



# User Manual

## Giganetix Camera Family

**GiGE**<sup>®</sup>  
VISION

**GEN< i >CAM**

### **For customers in the U.S.A.**

The equipment provided in an enclosure / housing has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

Modifications not expressly approved in this manual could void the user's authority to operate the equipment under FCC rules.

### **For customers in Canada**

This apparatus complies with the Class A limits for radio noise emissions set out in the Radio Interference Regulations.

### **Pour utilisateurs au Canada**

Cet appareil est conforme aux normes classe A pour bruits radioélectriques, spécifiées dans le Règlement sur le brouillage radioélectrique.

### **Life support applications**

These products are not designed for use in life support systems, appliances or devices where malfunction of the products can reasonably be expected to result in personal injury. Customers, Integrators and End Users using or selling these products for use in such applications do so at their own risk and agree to fully indemnify SMARTEK d.o.o. for any damages resulting from any improper use or sale.

## **Trademarks**

All trademarks, trade names and products represented in this document, unless stated otherwise, are brands protected internationally by law. No use of these may be made without prior, written authorization of SMARTEK d.o.o except to identify the products or services of the company.

## **Warranty**

SMARTEK d.o.o. has made reasonable efforts to ensure that the information provided in this document is accurate at the time of inclusion. However there may be unintentional and occasional errors for which we apologize. SMARTEK d.o.o. makes no representations, warranties or assurances of any kind as to the accuracy, currency or completeness of the information provided. SMARTEK d.o.o. shall not be liable of any damages or injury resulting from your reliance on any information provided in this document.

## **Copyright**

All texts, pictures and graphics and intellectual property in this document are protected by copyright. Reproduction of part or all of the content for trade or transfer purposes is prohibited. None of the content of this document may be copied or otherwise incorporated into or stored in any other website, electronic retrieval system, publication or other work in any form (whether hard copy, electronic or other). For the avoidance of doubt, framing of this document or any part of it is not permitted without express permission.

## Table of Contents

<b>1</b>	<b>Description of Product Family</b>	<b>1</b>
1.1	Precautions	2
1.2	Supported Industry Standards	3
1.2.1	GigE Vision	3
1.2.2	GenICam	4
1.2.3	C-Mount	5
1.3	EMI and ESD Consideration	5
1.4	Supported Third-Party Software	6
<b>2</b>	<b>SMARTEK Vision Giganetix Camera Models</b>	<b>7</b>
2.1	Mechanical and Electrical Specifications	8
2.1.1	Giganetix with Standard Housing (GC Series)	8
2.1.2	Giganetix with 90° Angled Housing (GC-S90 Series)	10
2.1.3	Giganetix Board Level (GC-BL Series)	12
2.1.4	Giganetix Plus Camera with Standard Housing (GCP Series)	17
2.2	Sensor Information and Technical Specification (All Models Separate)	19
2.2.1	GC1281M	19
2.2.2	GC2041C	21
2.2.3	GC2591M / GC2591C	23
2.2.4	GC3851M / GC3851C	25
2.2.5	GC1932M / GC1932C	27
2.2.6	GC651M / GC651C	29
2.2.7	GC652M / GC652C	31
2.2.8	GC653M / GC653C	33
2.2.9	GC781M / GC781C	35
2.2.10	GC1031M / GC1031C	37
2.2.11	GC1291M / GC1291C	39
2.2.12	GC1391M / GC1391C	41
2.2.13	GC1392M / GC1392C	43
2.2.14	GC1621M / GC1621C	45
2.2.15	GC2441M / GC2441C	47
2.2.16	GC1021M / GC1021C	49
2.2.17	GC1601M / GC1601C	51
2.2.18	GC1921M / GC1921C	53
2.2.19	GCP1931M / GCP1931C	55
2.2.20	GCP2061M / GCP2061C	57
2.2.21	GCP2461M / GCP2461C	59
2.2.22	GCP1941M / GCP1941C	61
2.2.23	GCP2751M / GCP2751C	63
2.2.24	GCP3381M / GCP3381C	65
2.2.25	GCP4241M / GCP4241C	67
2.3	Physical Interfaces	69
2.3.1	Ethernet Interface	69
2.3.2	Power and I/O-Interface	71
2.3.3	Temperature Specification and Heat Dissipation	77
2.3.4	IR-Cut Filter	79
2.3.5	Ingress Protection Class	79
2.4	Declarations of Conformity	80



2.4.1	CE	80
2.4.2	FCC	81
2.4.3	RoHS II	82
2.5	List of Supported Features	83
<b>3</b>	<b>Smartek GigEVisionSDK Library</b>	<b>85</b>
3.1	Supported Operating Systems	85
3.2	(Un-)Installing the GigEVisionSDK on Microsoft Windows and Linux	86
3.3	Unattended Installation on Microsoft Windows Operating Systems	86
3.4	Manual Filter Driver Installation / Uninstallation	87
3.5	User Buffer	88
3.6	GigEVisionClient	89
3.6.1	Graphical User Interface (GUI)	89
3.6.2	Acquire Images from Camera(s)	91
3.6.3	API Settings Dialog	97
3.6.4	Chunk Data Control	98
3.6.5	Log Dialog	100
3.6.6	Firmware Update	100
<b>4</b>	<b>Image Acquisition</b>	<b>102</b>
4.1	General Camera Architecture	102
4.1.1	CCD Sensor Readout	104
4.1.2	Multi-Tap CCD Sensor Readout	105
4.1.3	CMOS Sensor Readout	107
4.1.4	CCD vs. CMOS - Sensor Performance	108
4.1.5	Color Imaging with Bayer Pattern	109
4.2	Shutter types and Frame Readout	111
4.2.1	Global Shutter Readout	111
4.2.2	Electronic Rolling Shutter (ERS) Readout	112
4.2.3	Global Reset Release (GRR) Readout	114
4.3	Brightness and Sensor Signal Control	115
4.3.1	Exposure / Integration Time	115
4.3.2	Analog Gain and Black Level	117
4.3.3	Automatic Exposure and Gain Control	120
4.3.4	Automatic Tap Balancing	121
4.3.5	Digital Shift	123
4.4	Region of Interest (ROI)	124
4.4.1	Multiple Regions of Interest	125
4.4.2	Region of Interest Centering	127
4.5	Acquisition Control	128
4.5.1	Free Run Operation	129
4.5.2	Triggered Operation	130
4.6	Digital Input / Output Control	133
4.6.1	Input Lines	133
4.6.2	Output Lines	136
<b>5</b>	<b>Image Transmission over Gigabit Ethernet</b>	<b>137</b>
5.1	Smartek GigE Vision Filter Driver	137
5.1.1	UDP Packet Resend Mechanism	137
5.1.2	Inter Packet Delay	143
5.1.3	Frame Transfer Delay	145

---

5.2	LAN IP Configuration . . . . .	147
5.3	Network Interface Optimization . . . . .	148
5.3.1	Choosing the right Network Interface Card . . . . .	148
5.3.2	Using Jumbo Frames / Packets . . . . .	148
5.3.3	Raising Receive Buffers . . . . .	150
5.3.4	Disable the Interrupt Moderation Rate . . . . .	151
5.3.5	Disable the Flow Control . . . . .	152
5.4	Digital Image and Pixel Formats . . . . .	153
5.4.1	Image Layout . . . . .	153
5.4.2	Supported Pixel Formats for Monochrome Cameras . . . . .	154
5.4.3	Supported Pixel Formats for Color Cameras . . . . .	156
5.4.4	Pixel Formats Supported by the SMARTEK Vision Giganetix camera family . . . . .	158
5.5	Chunk Data . . . . .	161
5.5.1	Getting Started with Chunk Data . . . . .	162
5.5.2	Additional chunks . . . . .	165
<b>6</b>	<b>Image Processing on Camera</b>	<b>167</b>
6.1	Luminance Look-up Table . . . . .	167
6.2	Gamma Adjustment . . . . .	169
<b>7</b>	<b>Image Processing in GigEVisionSDK</b>	<b>170</b>
7.1	Image Statistics . . . . .	171
7.1.1	Histogram . . . . .	171
7.1.2	Average Luminance Calculation . . . . .	175
7.2	Image Processing Algorithms . . . . .	177
7.2.1	Luminance Look-Up Table (LUT) . . . . .	177
7.2.2	Digital Gain . . . . .	183
7.2.3	Auto Exposure and Auto Gain . . . . .	186
7.2.4	White Balance . . . . .	188
7.2.5	Gamma Correction . . . . .	191
7.2.6	Color Filter Array Interpolation (Demosaicing / Debayering) . . . . .	194
7.2.7	Matrix Multiplication 3x3 . . . . .	198
7.2.8	GIMP HSL . . . . .	200
7.2.9	Sharpening . . . . .	203
7.2.10	RGB to Grayscale Conversion . . . . .	205
7.2.11	Bit Depth Conversion . . . . .	207
7.2.12	Flip / Rotate Transformation . . . . .	208
7.3	Color Image Processing Pipeline . . . . .	210
<b>8</b>	<b>Contact Information</b>	<b>211</b>
<b>9</b>	<b>Revision History</b>	<b>212</b>

## 1 Description of Product Family

The SMARTEK Vision Giganetix camera family offers an affordable, easy to use set of digital cameras designed to meet demanding high quality image machine vision applications conforming to the industrial GigE Vision standard. The compact housings fit almost every space critical application. A wide selection of Sony, Aptina and Truesense Imaging CCD and CMOS sensors delivers images with high sensitivity and low noise. Excellent price to performance ratio makes this portfolio the perfect choice for every demanding user.

SMARTEK Vision Giganetix cameras combine standard Gigabit Ethernet technology with the GigEVisionSDK image acquisition software to reliably capture and transfer images from the camera to the PC. All Giganetix cameras are supported by one Software Development Kit as well as a large number of 3rd-party libraries compliant to the GigE Vision Standard. To use these devices with other software than provided by SMARTEK Vision, please check their user guides.

Ultra small compact form	Precise image sensor alignment
Sony, Aptina and Truesense Imaging CCD and CMOS sensors	Built-in IR cut-off filter in color models (optional for monochrome models)
Long cable length up to 100m	Standard C-Mount lens adapter
Use of low cost Cat5e or Cat6 Ethernet cables	90° angled and board level versions
Low power consumption, low thermal dissipation	Excellent thermal linkage between sensor and housing
Pixel depth of up to 14bit	Horizontal and vertical binning*
Very small trigger latency 2μs, jitter < 0.5μs	Opto-isolated inputs and outputs
Partial scan and region of interest functions*	Very competitive price to performance ratio
High frame rates or high sensitivity option*	Firmware update via SDK over Ethernet
Black anodized aluminum housing	Rubber sealed image sensor space
Internal image buffer for retransmission and reliability (packet resend mechanism)	Industrial connectors: EIAJ (Hirose)-12 pin and screw mount RJ45

\* model specific feature

Table 1: Key Benefits and Features

## 1.1 Precautions



Due to the ultra-small compact housing of the camera, it has a tendency to develop a high temperature. To maintain an optimal working temperature, mount the camera on a metal surface.



Do not attempt to disassemble this camera, there are sensitive optical parts inside. Tampering with it could lead to permanent damage.



Do not expose this camera to rain or moisture. This device is not intended to work under wet conditions.



Do not face this camera towards the sun, extremely bright light or light reflecting objects. Even when the camera is not in use, put the supplied lens cap on the lens mount, to prevent damage to the sensor



Handle this camera with the maximum care. Do not throw the device; there are fragile glass parts inside.



Operate this cameras only with the type of power source that meets the specifications indicated on the camera and within the documentation. Operating the camera outside of the specifications can cause to permanent damage. Further electrical specifications can be found in chapter 2.1 - Mechanical and Electrical specifications.

## 1.2 Supported Industry Standards

### 1.2.1 GigE Vision

*GigE Vision* is a communication interface standard for high-performance industrial cameras based on the Gigabit Ethernet technology. The main idea driving the development of the standard is to unify different protocols used in machine vision industrial applications and make hardware and software from various vendors interoperate seamlessly over GigE connections. GigE Vision is administered by the *Automated Imaging Association* (AIA).



#### Features of the GigE Vision standard:

- Fast data transfer rates - up to 1 Gbit/s (based on 1000BASE-T)
- Data transfer length up to 100m exceeding maximum length of FireWire, USB and Camera Link interfaces.
- Based on established standard allowing communication with other Ethernet devices and computers.

#### GigE Vision has four main elements:

- *GigE Vision Control Protocol* (GVCP) - runs on the UDP protocol. The standard defines how an application controls and configures devices, and instantiates stream channels on the device. It also defines the way for the device to notify an application about specific events.
- *GigE Vision Stream Protocol* (GVSP) - covers the definition of data types and the ways images and other data are transferred from device to application.
- *GigE Device Discovery Mechanism* - provides mechanisms for a device to obtain valid IP address and for an application to enumerate devices on the network.
- *XML description* - file based on the GenICam standard which provides the mapping between a device feature and the device register implementing the feature.

### 1.2.2 GenICam

*GenICam* (Generic Interface for Cameras) is a generic programming interface for machine vision cameras. The goal of the standard is to decouple industrial camera interface technology (such as GigE Vision, Camera Link, USB or FireWire) from the user application programming interface (API). GenICam is administered by the *European Machine Vision Association* (EMVA).



GenICam consists of three modules to help solve the main tasks in machine vision field in a generic way. These modules are:

- *GenApi* - configures the camera and details how to access and control cameras by using an XML description file.
- *Standard Feature Naming Convention* (SFNC) - are the recommended names and types for common features in cameras to promote interoperability
- *GenTL* - is the transport layer interface for enumerating cameras, grabbing images from the camera, and moving them to the user application.

#### GenICam provides supports for five basic functions:

- Configuring the camera - supports a range of camera features such as frame size, acquisition speed, pixel format, gain, image offset, etc.
- Grabbing images - creates access channels between the camera and the user interface and initiates receiving images.
- Graphical user interface - enables user GUI interface to seamlessly talk to the camera(s).
- Transmitting extra data - enables cameras to send extra data on top of the image data. Typical examples could be histogram information, time stamp, area of interest in the frame, etc.
- Delivering events - enables cameras to talk to the application through an event channel

#### Standard Features Naming Convention (SFNC)

SFNC provides the definitions of standard use cases and standard features. The goal is to cover and to standardize the naming convention used in all those basic use cases where the implementation by different vendors would be very similar anyway. The GenICam technology allows exposing arbitrary features of a camera through a unified API and GUI. Each feature can be defined in an abstract manner by its name, interface type, unit of measurement and behavior. The GenApi module of the GenICam standard defines how to write a camera description file that describes a specific camera's mapping.

For detailed information about this convention visit [www.emva.org](http://www.emva.org).

### 1.2.3 C-Mount

A *C-Mount* is a type of lens mount commonly found on 16mm movie cameras, closed-circuit television cameras (CCTV), trinocular microscope photo tubes and CCD/CMOS digital cameras. C-Mount lenses provide a male thread which mates with a female thread on the camera. The thread is nominally 25.4mm [1"] in diameter, with 32 threads per inch, designated as "1-32 UN 2A" in the ANSI B1.1 standard for unified screw threads. The flange focal distance is 17.526mm [0.69"] and thread length 3.8mm [0.15"].

## 1.3 EMI and ESD Consideration

Excessive EMI and ESD can cause problems with your camera such as false triggering or can cause the camera to suddenly stop capturing images. EMI and ESD can also have a negative impact on the quality of the image data transmitted by the camera.

To avoid problems with EMI and ESD, you should follow these general guidelines:

- Use high quality shielded cables. The use of high quality cables is one of the best defenses against EMI and ESD.
- Try to use camera cables with correct length and try to run the camera cables and power cables parallel to each other. Avoid coiling camera cables.
- Avoid placing camera cables parallel to wires carrying high-current, switching voltages such as wires supplying stepper motors or electrical devices that employ switching technology.
- Attempt to connect all grounds to a single point, e.g. use a single power outlet for the entire system and connect all grounds to the single outlet.
- Use a line filter on the main power supply.
- Install the camera and camera cables as far as possible from devices generating sparks.
- Decrease the risk of electrostatic discharge by taking the following measures:
  - Use conductive materials at the point of installation.
  - Use suitable clothing (cotton) and shoes.
  - Control the humidity in your environment. Low humidity can cause ESD problems.

## 1.4 Supported Third-Party Software

The Giganetix cameras have been verified to be applicable with the third-party software shown below in Table 2.

Software	Requirements
Cognex Vision Pro	Native (GigEVision interface)
Matrox Imaging Library	Native (GigEVision interface)
MVtec Halcon	Native (GigEVision interface)
National Instruments LabView	National Instruments IMAQdx (Plugin)
Scorpion Vision	Plugin provided by SMARTEK Vision

*Table 2: Third-Party Software*



## 2 SMARTEK Vision Giganetix Camera Models

The Giganetix camera family consists of a line-up of GigE Vision compliant cameras equipped with a selection of CCD and CMOS sensors, fitted into several different camera designs. The following chapter contains the hardware specification of the single camera series and their different models, including technical drawings. Table 3 gives a brief overview about the unique characteristics of each series.

Type	Short Description
<b>GC</b> (standard housing)	Standard Giganetix Camera
<b>GC-S90</b> (angled 90° housing)	Standard Giganetix Camera with 90° angled housing. For applications with very limited space in optical axis and other mechanical restrictions, making the standard housing unsuitable.
<b>GC-BL</b> (board level)	90° Board level version of the standard Giganetix Camera with a single mainboard and detached sensor head. Suitable for OEM and special solutions where a camera needs to be integrated into a closed device and/or a housed camera does not fit into the design.
<b>GCP</b> (standard housing)	Enhanced version of the Giganetix Camera in an adapted mechanical design providing a set of high-end sensors and increased hardware capabilities, as well as Power over Ethernet by default.

Table 3: Giganetix Family Camera Lines

## 2.1 Mechanical and Electrical Specifications

### 2.1.1 Giganetix with Standard Housing (GC Series)

The Giganetix camera series with standard housing represents the regular camera design for the GC series with the main focus on a small form factor, offering the comprehensive camera electronics in a small 35x35x48 mm footprint. Figure 1 shows an image of the housing. Table 4 contains an overview about the model specific specifications.



Figure 1: Giganetix Camera with Standard Housing

External dimensions (H x W x L)	35 x 35 x 48 [mm]	1.38 x 1.38 x 1.89 [in]
Housing	Black anodized aluminum case	
Weight	Approx. 90g	3.2oz
Storage temperature <sup>1</sup>	-30°C to +60°C	-22°F to +140°F
Operating temperature <sup>1</sup>	0°C to +50°C	+32°F to +122°F
Operating humidity	20% to 80%, relative, non-condensing	
Storage humidity	20% to 80%, relative, non-condensing	
Power requirement	10V to 24V DC via Power and I/O-interface, Power over Ethernet (PoE)	
Lens mount	C-Mount	
Connectors	Screw mount Ethernet RJ45 (Communication and Data), Circular Hirose 12 pin (Power and I/O-Interface)	
Digital input	2 input channels, opto-isolated	
Digital output	2 output channels, opto-isolated	
Conformity	CE, FCC, RoHS II, GigE Vision, GenICam, PoE (IEEE802.3af)	

<sup>1</sup> measured at camera housing

Table 4: Mechanical and electrical specifications

### 2.1.1.1 Technical Drawings

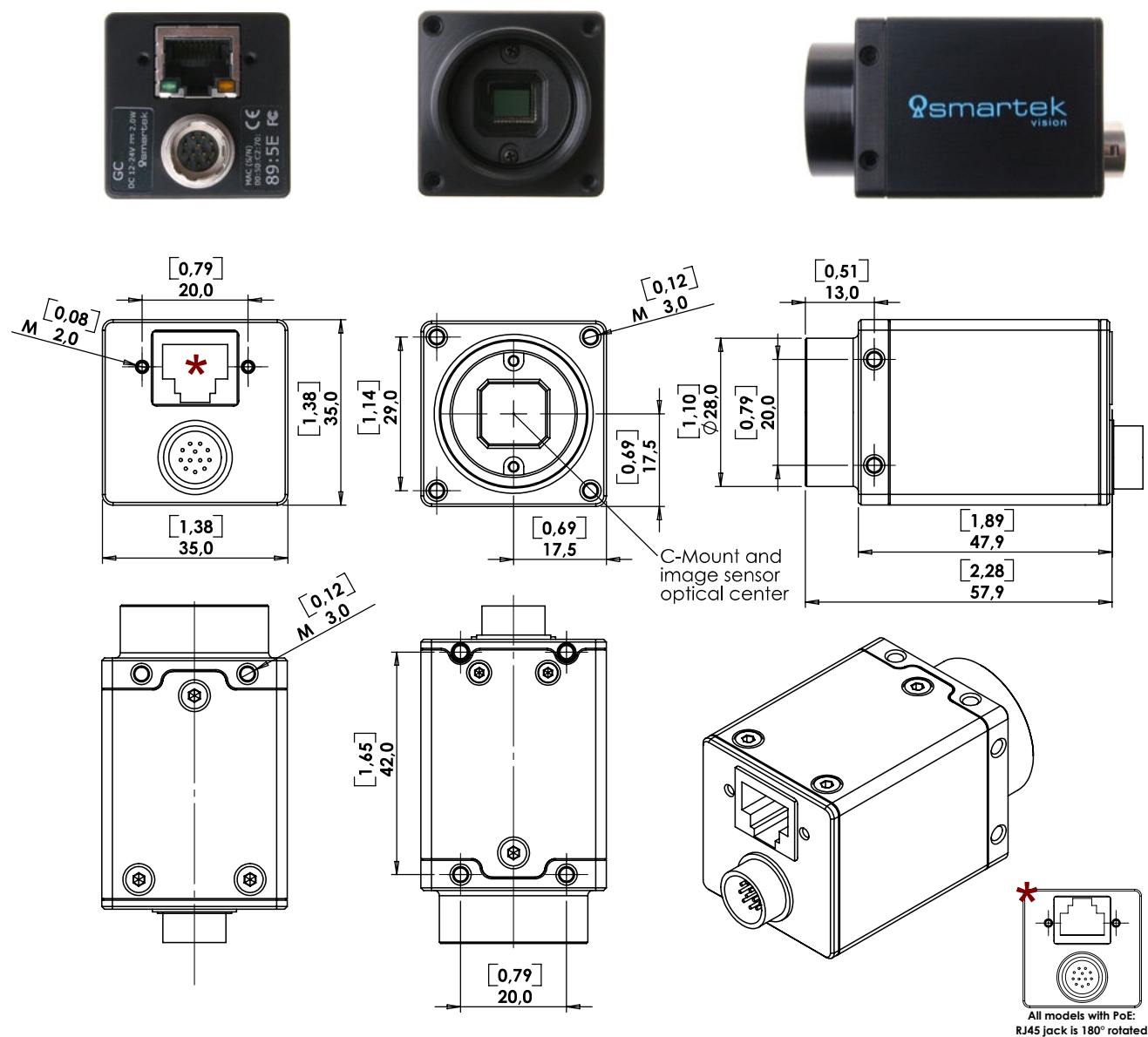


Figure 2: Technical measures of standard camera housing (all dimensions are in mm [inch])

### 2.1.2 Giganetix with 90° Angled Housing (GC-S90 Series)

The 90° angled version of the Giganetix camera series is identical to the standard GC camera, fitted into a different form factor to enhance its flexibility. It aims on very build-up applications and all such with limited space in sensor axis. Due to its non-symmetrical housing the GC-S90 series supports image mirroring to reverse the image about its X- and Y-Axis to allow a most possible flexibility in positioning.



Figure 3: Giganetix Camera with 90° angled housing

External dimensions (H x W x L)	35 x 35 x 75 [mm]	1.38 x 1.38 x 2.95 [in]
<b>Housing</b>	Black anodized aluminum case with 90° angled sensor and lens mount	
<b>Weight</b>	Approx. 120g	approx. 4.2oz
<b>Storage temperature<sup>1</sup></b>	-30°C to +60°C	-22°F to +140°F
<b>Operating temperature<sup>1</sup></b>	0°C to +50°C	+32°F to +122°F
<b>Operating humidity</b>	20% to 80%, relative, non-condensing	
<b>Storage humidity</b>	20% to 80%, relative, non-condensing	
<b>Power requirement</b>	10V to 24V DC via Power and I/O-interface	
<b>Lens mount</b>	C-Mount	
<b>Connectors</b>	Screw mount Ethernet RJ45 (Communication and Data), Circular Hirose 12 pin (Power and I/O-Interface)	
<b>Digital input</b>	2 input channels, opto-isolated	
<b>Digital output</b>	2 output channels, opto-isolated	
<b>Conformity</b>	CE, FCC, RoHS II, GigE Vision, GenICam	

<sup>1</sup> measured at camera housing

Table 5: Mechanical and electrical specifications

### 2.1.2.1 Technical Drawings

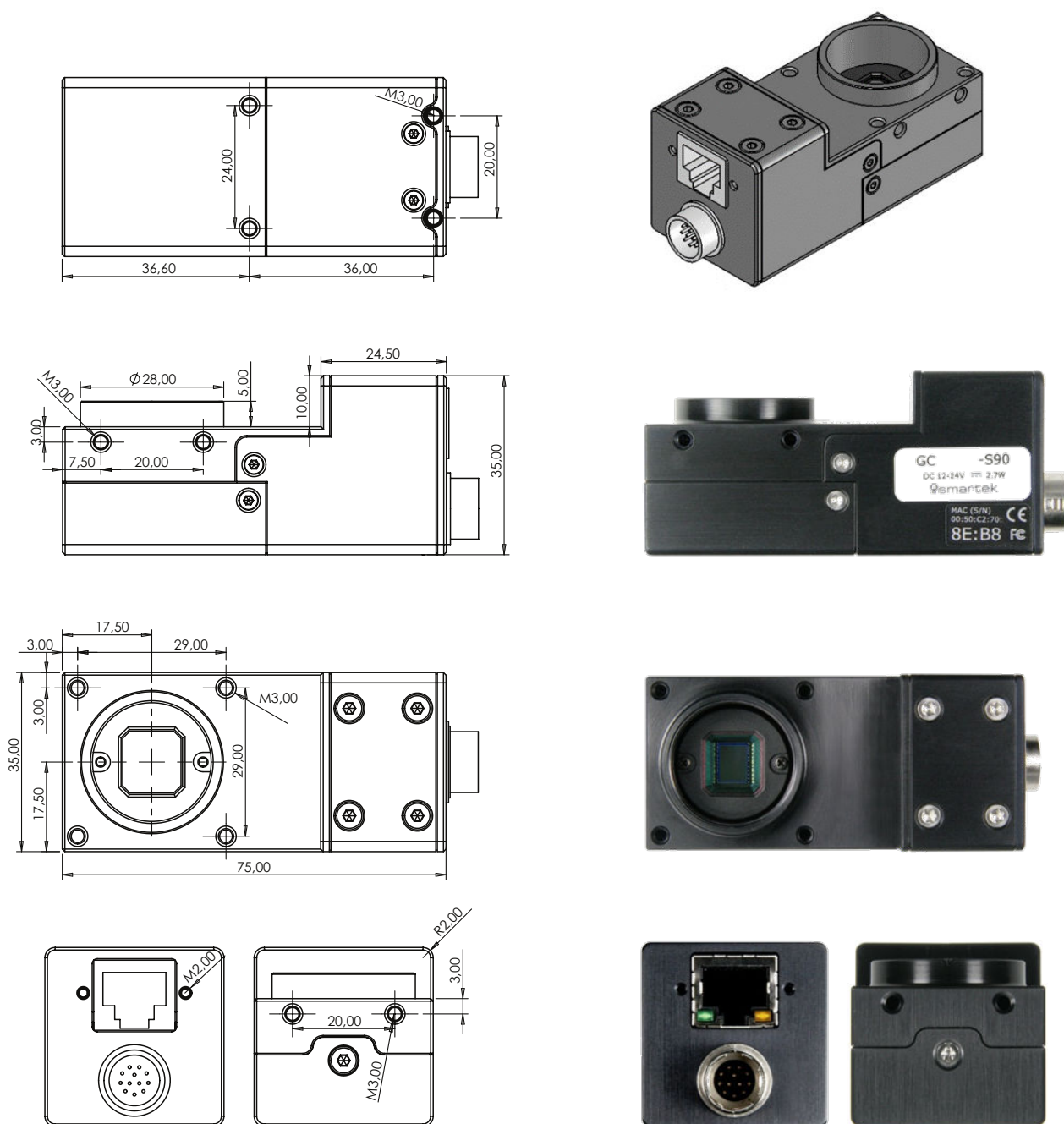


Figure 4: Technical measures of angled 90° camera housing (all dimensions are in mm [inch])

### 2.1.3 Giganetix Board Level (GC-BL Series)

The board level version of the Giganetix camera series aims on the OEM integration of the camera into closed customer devices. It provides the complete electrical design of the GC mainboard on a single board, having a separated sensor head supporting cable lengths of up to 150 mm.

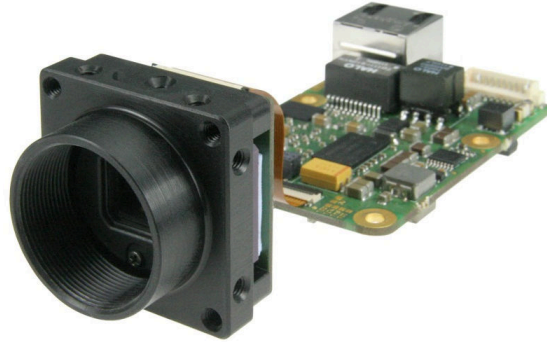


Figure 5: Giganetix Board Level Camera

<b>External dimensions (H x W x L)</b>	35 x 35 x 26.2 [mm]	Sensor board	1.38 x 1.38 x 1.03 [in]
	65 x 43 x 19 [mm]	Head board	2.56 x 1.69 x 0.75 [in]
<b>Housing</b>	No housing sensor head and mainboard only, connected via FPC cable		
<b>Weight</b>	Approx. 60g		approx. 2.1oz
<b>Storage temperature<sup>1</sup></b>	-30°C to +60°C		-22°F to +140°F
<b>Operating temperature<sup>1</sup></b>	0°C to +45°C		+32°F to +113°F
<b>Operating humidity</b>	20% to 80%, relative, non-condensing		
<b>Storage humidity</b>	20% to 80%, relative, non-condensing		
<b>Power requirement</b>	10V to 24V DC via Power and I/O-interface, Power over Ethernet (PoE)		
<b>Lens mount</b>	C-Mount		
<b>Connectors</b>	Screw mount Ethernet RJ45 (Communication and Data), Circular Hirose 12 pin (Power and I/O-Interface)		
<b>Digital input</b>	2 input channels, opto-isolated		
<b>Digital output</b>	2 output channels, opto-isolated		
<b>Conformity</b>	RoHS II, GigE Vision, GenICam, PoE (IEEE802.3af)		

<sup>1</sup> measured at the direct board environment

Table 6: Mechanical and electrical specifications

Equal to the GCP series, the board level version is equipped with the latest version of the camera's power supply and supports Power over Ethernet. Due to the large dimensioned components also sensors with an increased power consumption, like multi-tap CCDs, are supported.

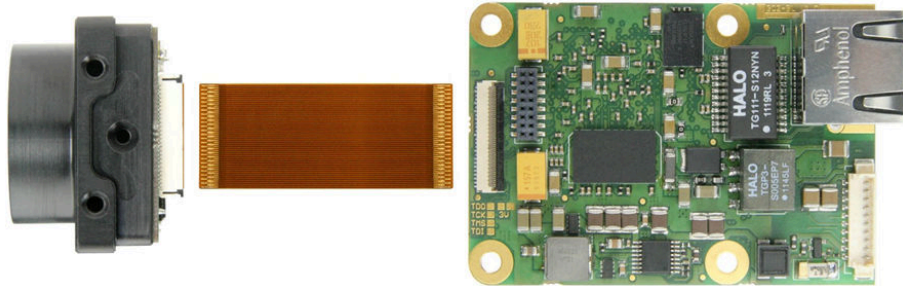


Figure 6: GC-BL - Sensor head, FPC cable and mainboard (from left to right)



### Note

For assembling instructions please refer to the GC-BL *Assemble Guide* available on [www.SMARTEKvision.com/downloads.php](http://www.SMARTEKvision.com/downloads.php) after reading the following safety instructions carefully.

### 2.1.3.1 Further Precautions for Board Level Cameras

Before the first operation of a Giganetix Board Level camera, please read the following safety instructions and cautions carefully. Abuse and misapplication may lead to limited or canceled warranty.

#### ESD Cautions:



All boards of the camera are sensitive to electrostatic discharge. Handle all components of the camera only in static-safe areas and make sure that no electrostatic loads from your body are discharged to any of the boards:

- Discharge yourself on a grounded body before touching
- Work in a static-safe work area on an antistatic mat
- Wear an antistatic-wrist strap the whole time handling the camera boards
- Do not hold any of the camera's components to you clothing

### General Cautions:



The board level cameras are delivered without a housing and partly disassembled. Handle all parts with care and do not touch the components or contacts on the boards; hold all boards only by their edges.



The cable used to connect sensor head and mainboard is a Flat Printed Circuit (FPC) cable. Due to the construction of cable and jack it is not build for re-plugging or multiple bending cycles; physical stress to cable or connector can lead to permanent damage.



Do not attempt to disassemble the lens mount or sensor head; there are sensitive optical parts inside, tampering can lead to permanent damage.



Building a case around the camera causes in a heat accumulation of the internal ambient temperature. Make sure that the environmental temperature of the camera electronics does not exceed the specified maximum.

### Environmental and Mechanical Cautions:



Due to the missing housing the camera is not certified to any EMC directives. The customer needs to take care of fulfilling EMC regulations for his individual target application and for sufficiently shielding the camera against environmental radiation.



Avoid the contact of the of the camera's boards with any liquid or any impurities; do only operate clean boards and protect them against environmental influences like particles, humidity, liquids and radiation by an appropriate protective housing.



Avoid any mechanical forces like torsion, tension and compression, e.g. by mounting the boards or the cabling. Make sure that no forces are induced to the connectors by using sufficient cable pull reliefs.



### 2.1.3.2 Technical Drawings

#### Sensor Head Dimensions (C-Mount):

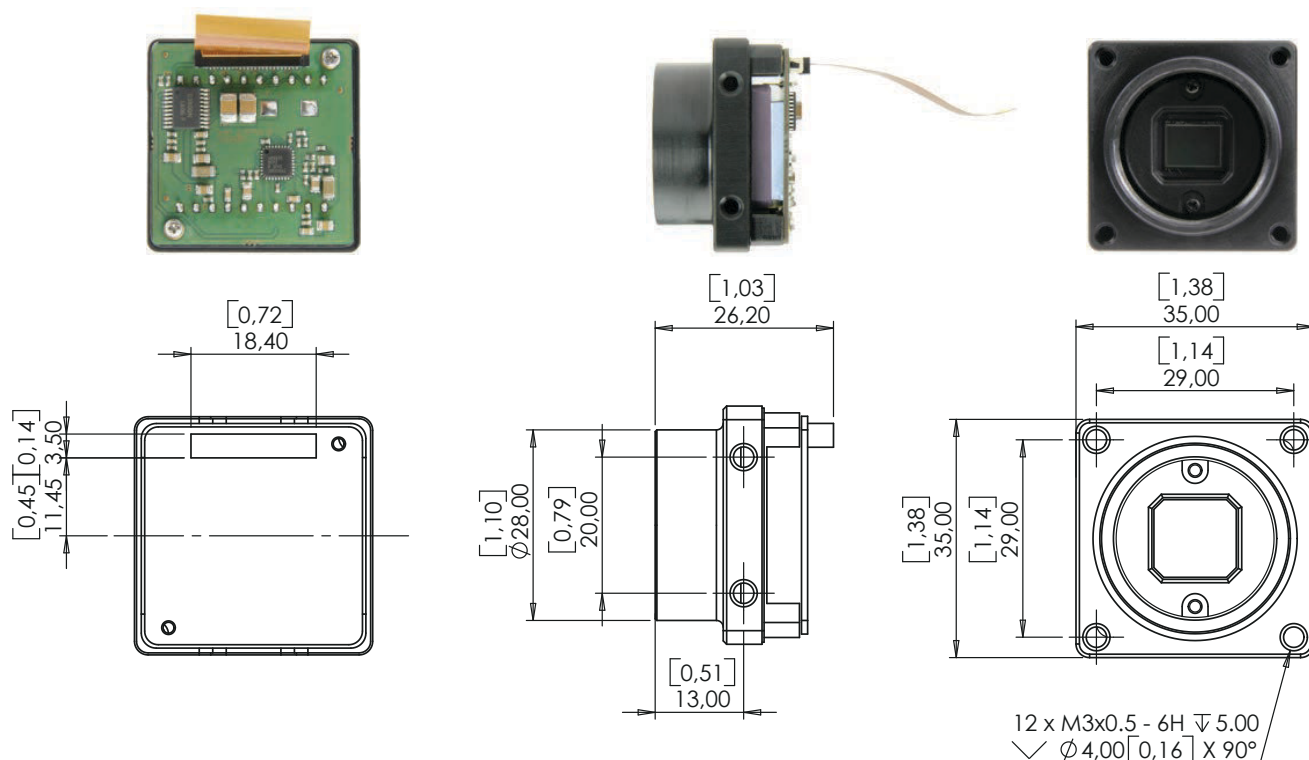


Figure 7: Technical measures of board level sensor head (all dimensions are in mm [inch])

### Mainboard Dimensions:

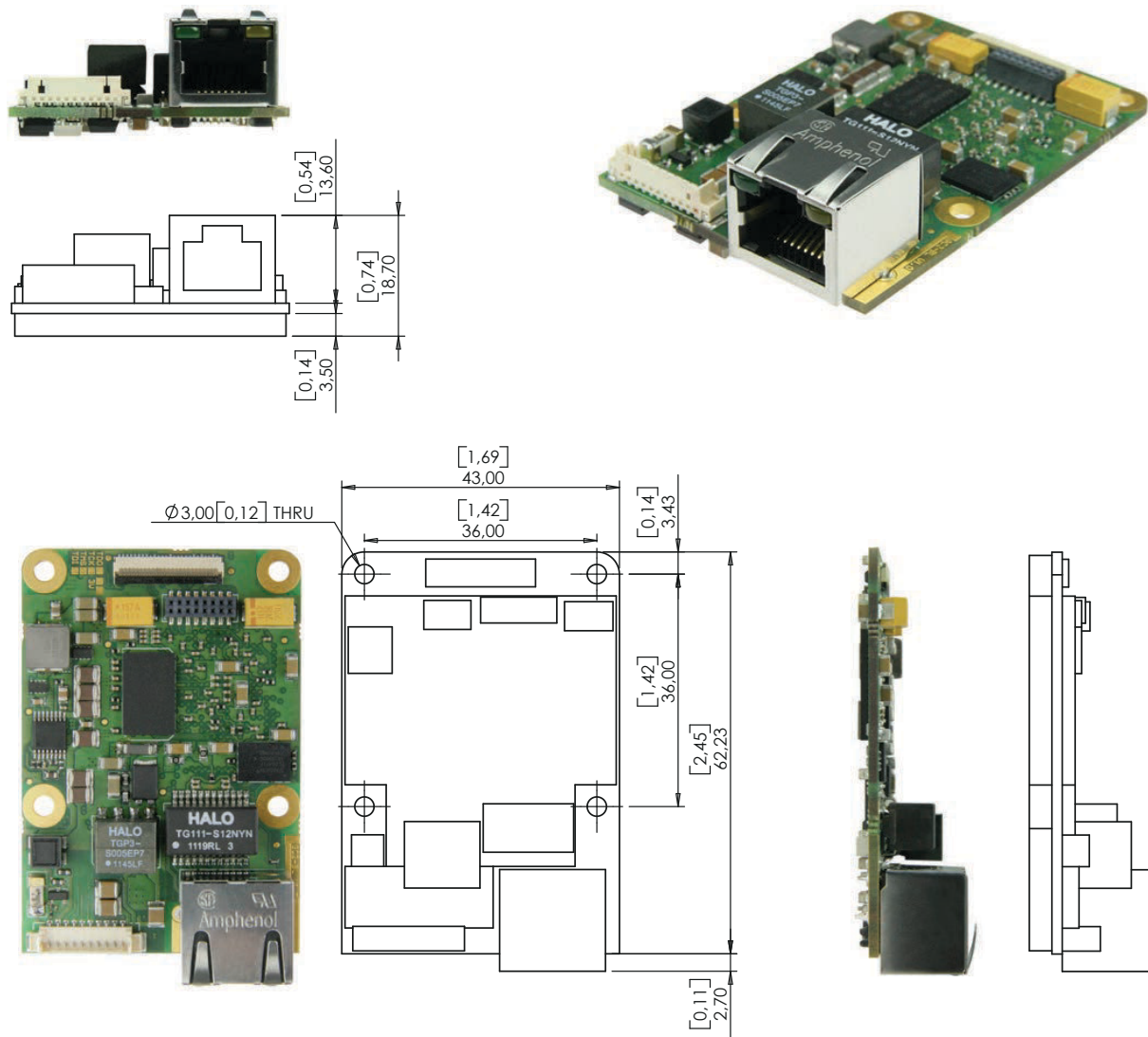


Figure 8: Technical measures of board level mainboard (all dimensions are in mm [inch])

### 2.1.4 Giganetix Plus Camera with Standard Housing (GCP Series)

The Giganetix Plus camera series is the enhanced version of the GC camera and allows with its extended hardware the integration of larger high-end sensors with higher data rates. Thanks to its well-spaced hardware it supports features like the factory based tap calibration and allows the integration of further image processing features, right up to fully customer specific implementations.



Figure 9: Giganetix Plus Camera with standard housing

External dimensions (H x W x L)	50 x 50 x 48 [mm]	1.97 x 1.97 x 1.89 [in]
<b>Housing</b>	Black anodized aluminum case	
<b>Weight</b>	Approx. 150g	5.3oz
<b>Storage temperature<sup>1</sup></b>	-30°C to +60°C	-22°F to +140°F
<b>Operating temperature<sup>1</sup></b>	0°C to +50°C	+32°F to +122°F
<b>Operating humidity</b>	20% to 80%, relative, non-condensing	
<b>Storage humidity</b>	20% to 80%, relative, non-condensing	
<b>Power requirement</b>	10V to 24V DC via Power and I/O-interface, Power over Ethernet (PoE)	
<b>Lens mount</b>	C-Mount	
<b>Connectors</b>	Screw mount Ethernet RJ45 (Communication, Data and Power), Circular Hirose 12 pin (Power and I/O-Interface)	
<b>Digital input</b>	2 input channels, opto-isolated	
<b>Digital output</b>	2 output channels, opto-isolated	
<b>Conformity</b>	CE, FCC, RoHS II, GigE Vision, GenICam, PoE (IEEE802.3af)	

<sup>1</sup> measured at camera housing

Table 7: Mechanical and electrical specifications

### 2.1.4.1 Technical Drawings

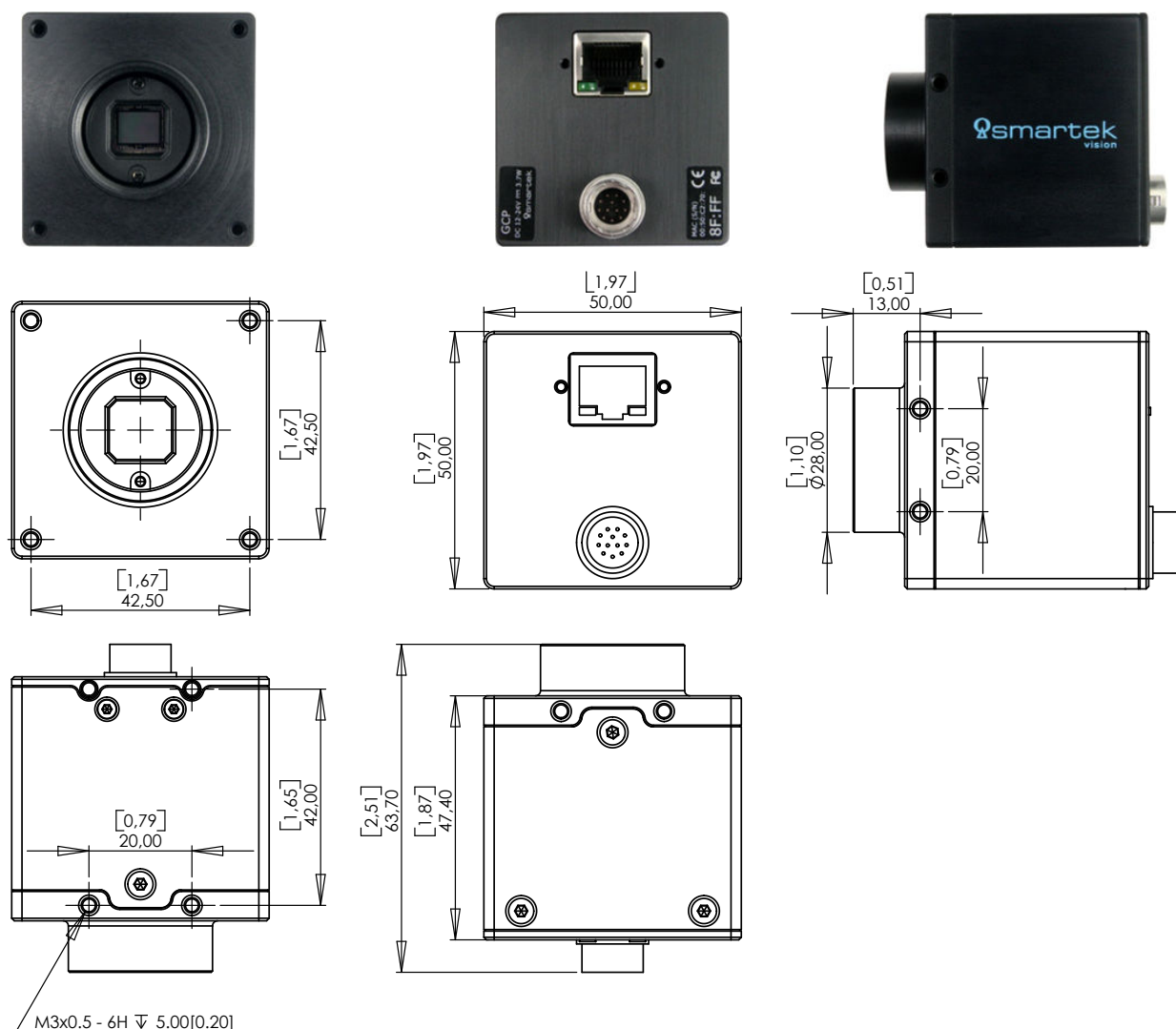


Figure 10: Technical measures of GCP camera with standard housing (all dimensions are in mm [inch])

## 2.2 Sensor Information and Technical Specification (All Models Separate)

The following chapter contains sensor specific specifications for all existing camera models, including the respective response curves. All response curves have been extracted from the datasheet of the sensor manufacturer.

### 2.2.1 GC1281M

	GC	GC-S90	GC-BL
<b>Image Sensor</b>	Aptina MT9M001		
<b>Chromatics</b>	Monochrome		
<b>Sensor type</b>	CMOS		
<b>Sensor resolution (H x W)</b>	1280 x 1024		
<b>Optical size</b>	1/2"		
<b>Pixel size (in <math>\mu\text{m}</math>)</b>	5.2 x 5.2		
<b>Analog gain (in dB)</b>	0 to 23.5		
<b>Shutter</b>	Rolling		
<b>Exposure time</b>	32 $\mu\text{s}$ to 0.5s		
<b>Max. frame rate (8Bit; in Hz)</b>	30		
<b>ADC bit depth</b>	8 bit		
<b>Pixel data formats</b>	Mono8		
<b>Synchronization</b>	Free run, external and software trigger (single shot, multi shot)		
<b>Exposure control</b>	Freely programmable via GigE Vision interface		
<b>Power consumption (aux. / 12V)</b>	2.3W	2.3W	2.3W
<b>Power consumption (PoE)</b>	3.0W	Not supported	3.0W

Table 8: Model specific specification of GC1281M

## Relative Response

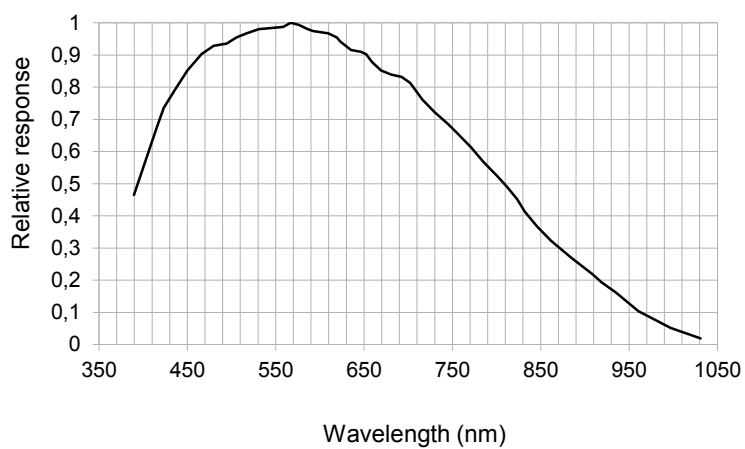


Figure 11: Relative response of GC1281 Monochrome (from sensor datasheet)

### 2.2.2 GC2041C

	GC	GC-S90	GC-BL
<b>Image Sensor</b>	Aptina MT9T031		
<b>Chromatics</b>	Color		
<b>Sensor type</b>	CMOS		
<b>Sensor resolution (H x W)</b>	2048 x 1536		
<b>Optical size</b>	1/2"		
<b>Pixel size (in <math>\mu\text{m}</math>)</b>	3.2 x 3.2		
<b>Analog gain (in dB)</b>	0 to 23.5		
<b>Shutter</b>	Rolling		
<b>Exposure time</b>	53 $\mu\text{s}$ to 10s		
<b>Max. frame rate (8Bit; in Hz)</b>	12		
<b>ADC bit depth</b>	8 bit		
<b>Pixel data formats</b>	Mono8, BayerGR8		
<b>Synchronization</b>	Free run, external and software trigger (single shot, multi shot)		
<b>Exposure control</b>	Freely programmable via GigE Vision interface		
<b>Power consumption (aux. / 12V)</b>	2.2W	2.2W	2.2W
<b>Power consumption (PoE)</b>	2.8W	Not supported	2.8W

Table 9: Model specific specification of GC2041C

## Relative Response

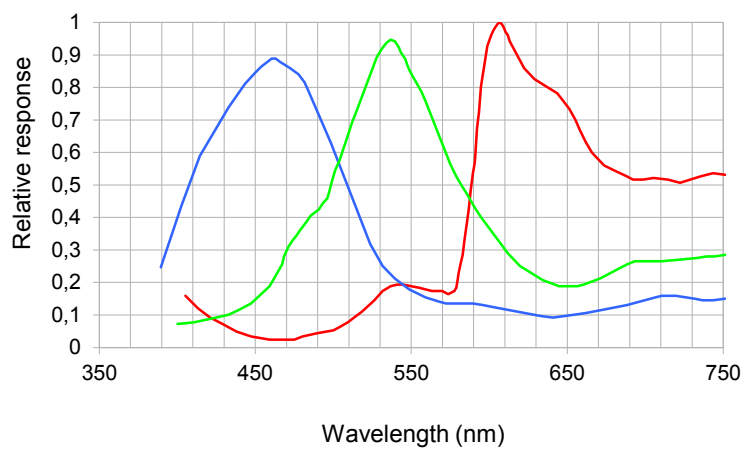


Figure 12: Relative response of GC2041 Color (from sensor datasheet)



### 2.2.3 GC2591M / GC2591C

	GC	GC-S90	GC-BL
<b>Image Sensor</b>	Aptina MT9P031		
<b>Chromatics</b>	Monochrome, Color		
<b>Sensor type</b>	CMOS		
<b>Sensor resolution (H x W)</b>	2592 x 1944		
<b>Optical size</b>	1/2.5"		
<b>Pixel size (in <math>\mu\text{m}</math>)</b>	2.2 x 2.2		
<b>Analog gain (in dB)</b>	0 to 23.5		
<b>Shutter</b>	Rolling		
<b>Exposure time</b>	36 $\mu\text{s}$ to 10s		
<b>Max. frame rate (8Bit; in Hz)</b>	14		
<b>ADC bit depth</b>	8 bit		
<b>Pixel data formats</b>	Mono8, BayerGR8		
<b>Synchronization</b>	Free run, external and software trigger (single shot, multi shot)		
<b>Exposure control</b>	Freely programmable via GigE Vision interface		
<b>Power consumption (aux. / 12V)</b>	2.2W	2.2W	2.2W
<b>Power consumption (PoE)</b>	3.0W	Not supported	3.0W

Table 10: Model specific specification of GC2591

## Relative Response

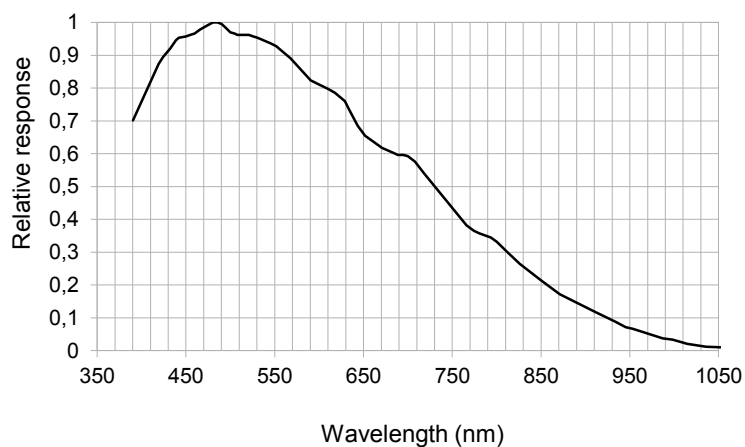


Figure 13: Relative response of GC2591 Monochrome (from sensor datasheet)

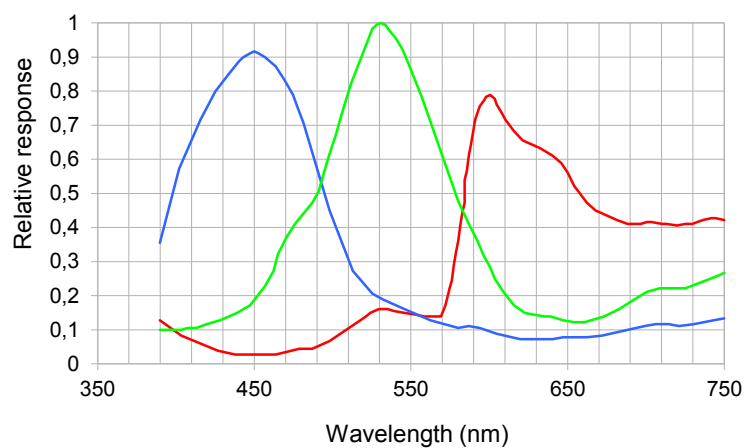


Figure 14: Relative response of GC2591 Color (from sensor datasheet)

### 2.2.4 GC3851M / GC3851C

	GC	GC-S90	GC-BL
<b>Image Sensor</b>	Aptina MT9J003		
<b>Chromatics</b>	Monochrome, Color		
<b>Sensor type</b>	CMOS		
<b>Sensor resolution (H x W)</b>	3856x2764	3848x2762	3856x2764
<b>Optical size</b>	1/2.3"		
<b>Pixel size (in <math>\mu\text{m}</math>)</b>	1.67 x 1.67		
<b>Analog gain (in dB)</b>	0 to 23.5		
<b>Shutter</b>	Rolling (free run), Global reset release (triggered)		
<b>Exposure time</b>	36 $\mu\text{s}$ to 10s		
<b>Max. frame rate (8Bit; in Hz)</b>	7		
<b>ADC bit depth</b>	8 bit		
<b>Pixel data formats</b>	Mono8, BayerGR8		
<b>Synchronization</b>	Free run, external and software trigger (single shot, multi shot)		
<b>Exposure control</b>	Freely programmable via GigE Vision interface		
<b>Power consumption (aux. / 12V)</b>	2.5W	2.5W	2.5W
<b>Power consumption (PoE)</b>	3.2W	Not supported	3.2W

Table 11: Model specific specification of GC3851

## Relative Response

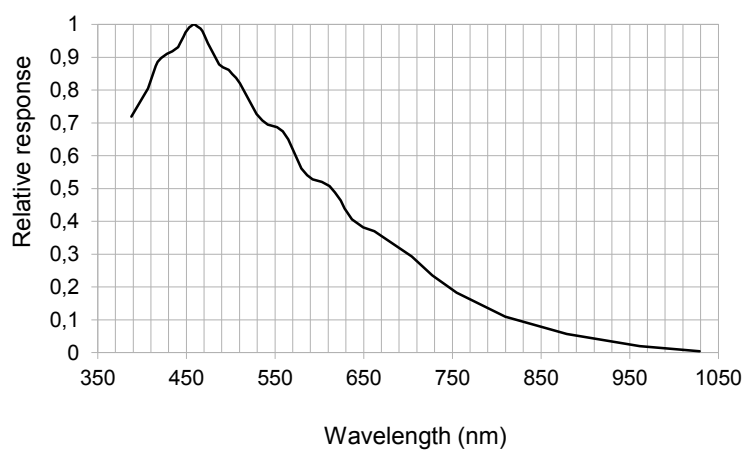


Figure 15: Relative response of GC3851 Monochrome (from sensor datasheet)

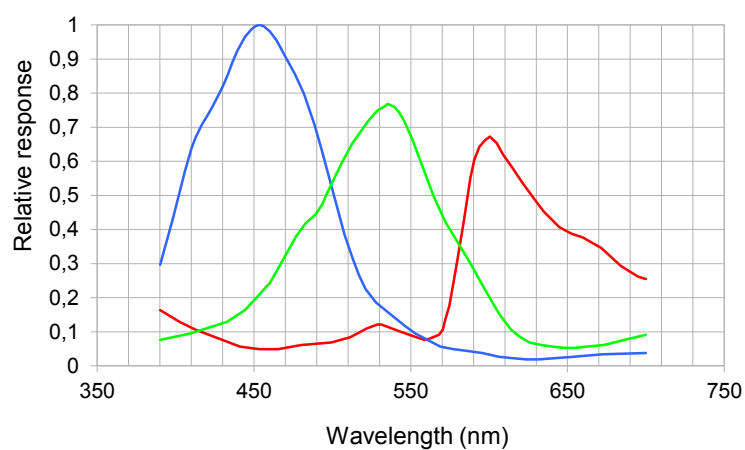


Figure 16: Relative response of GC3851 Color (from sensor datasheet)

### 2.2.5 GC1932M / GC1932C

	GC	GC-S90	GC-BL
<b>Image Sensor</b>	Sony IMX249		
<b>Chromatics</b>	Monochrome, Color		
<b>Sensor type</b>	CMOS		
<b>Sensor resolution (H x W)</b>	1936 x 1216		
<b>Optical size</b>	1/1.2"		
<b>Pixel size (in <math>\mu\text{m}</math>)</b>	5.86 x 5.86		
<b>Analog gain (in dB)</b>	0 to 24		
<b>Shutter</b>	Global Shutter		
<b>Exposure time</b>	26 $\mu\text{s}$ to 10s		
<b>Max. frame rate (8Bit; in Hz)</b>	41		
<b>ADC bit depth</b>	8bit, 10 bit		
<b>Pixel data formats (mono model)</b>	Mono8, Mono10Packed, Mono16		
<b>Pixel data formats (color model)</b>	Mono8, Mono16, BayerRG8, BayerRG16		
<b>Synchronization</b>	Free run, external and software trigger (single shot, multi shot)		
<b>Exposure control</b>	Freely programmable via GigE Vision interface		
<b>Power consumption (aux. / 12V)</b>	2.6W	2.6W	2.6W
<b>Power consumption (PoE)</b>	3.4W	Not supported	3.4W

Table 12: Model specific specification of GC1932

## Relative Response

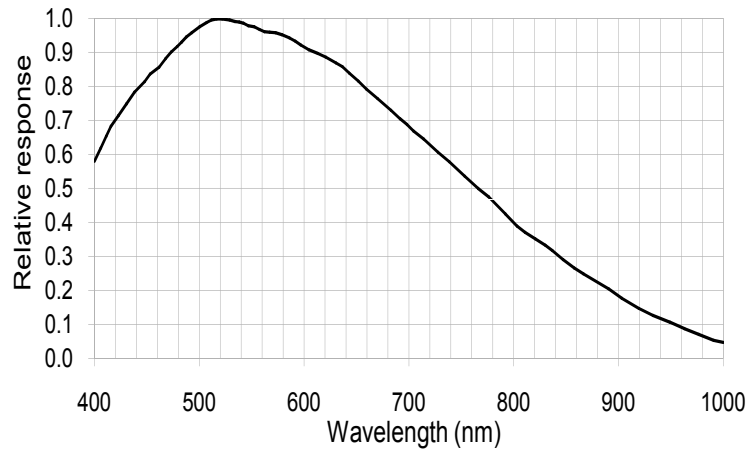


Figure 17: Relative response of GC1932 Monochrome (from sensor datasheet)

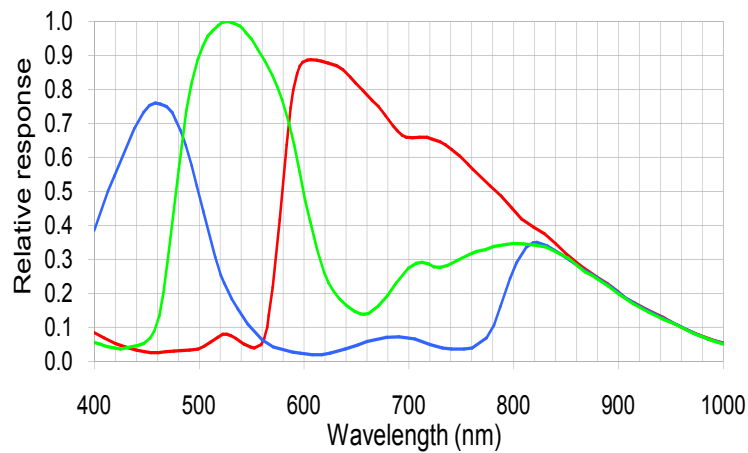


Figure 18: Relative response of GC1932 Color (from sensor datasheet)

### 2.2.6 GC651M / GC651C

	GC	GC-S90	GC-BL
<b>Image Sensor</b>	Sony ICX618		
<b>Chromatics</b>	Monochrome, Color		
<b>Sensor type</b>	CCD		
<b>Sensor resolution (H x W)</b>	659x494	656x492	659x494
<b>Optical size</b>	1/4"		
<b>Pixel size (in <math>\mu\text{m}</math>)</b>	5.6 x 5.6		
<b>Analog gain (in dB)</b>	5.1 to 41.8		
<b>Shutter</b>	Progressive Scan		
<b>Exposure time</b>	10 $\mu\text{s}$ to 10s		
<b>Max. frame rate (8Bit; in Hz)</b>	120		
<b>ADC bit depth</b>	8 bit, 14 bit		
<b>Pixel data formats (mono model)</b>	Mono8, Mono16		
<b>Pixel data formats (color model)</b>	Mono8, Mono16, BayerRG8, BayerRG16		
<b>Synchronization</b>	Free run, external and software trigger (single shot, multi shot)		
<b>Exposure control</b>	Freely programmable via GigE Vision interface		
<b>Power consumption (aux. / 12V)</b>	2.3W	2.3W	2.3W
<b>Power consumption (PoE)</b>	3.0W	Not supported	3.0W

Table 13: Model specific specification of GC651MC

## Relative Response

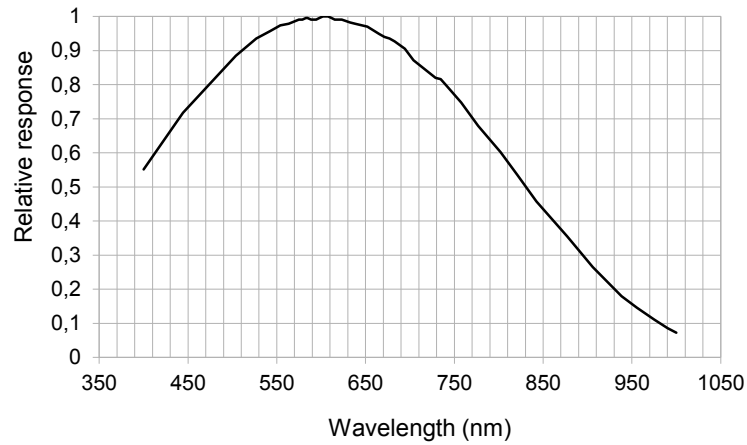


Figure 19: Relative response of GC651 Monochrome (from sensor datasheet)

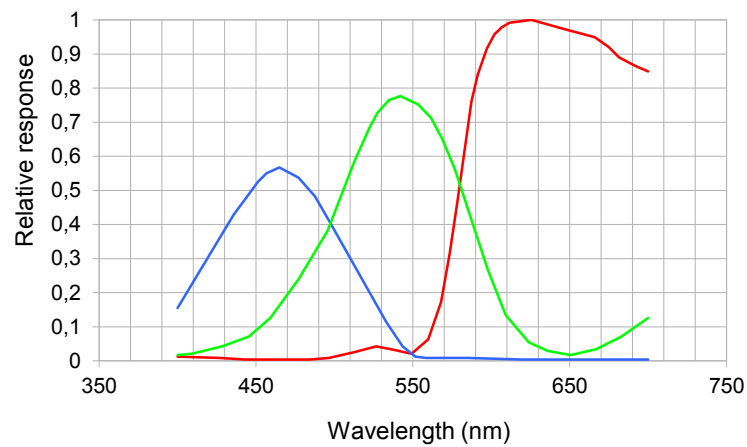


Figure 20: Relative response of GC651 Color (from sensor datasheet)



### 2.2.7 GC652M / GC652C

	GC	GC-S90	GC-BL
<b>Image Sensor</b>	Sony ICX424		
<b>Chromatics</b>	Monochrome, Color		
<b>Sensor type</b>	CCD		
<b>Sensor resolution (H x W)</b>	659x494	656x492	659x494
<b>Optical size</b>	1/3"		
<b>Pixel size (in <math>\mu\text{m}</math>)</b>	7.4 x 7.4		
<b>Analog gain (in dB)</b>	5.1 to 41.8		
<b>Shutter</b>	Progressive Scan		
<b>Exposure time</b>	10 $\mu\text{s}$ to 10s		
<b>Max. frame rate (8Bit; in Hz)</b>	97		
<b>ADC bit depth</b>	8 bit, 14 bit		
<b>Pixel data formats (mono model)</b>	Mono8, Mono16		
<b>Pixel data formats (color model)</b>	Mono8, Mono16, BayerRG8, BayerRG16		
<b>Synchronization</b>	Free run, external and software trigger (single shot, multi shot)		
<b>Exposure control</b>	Freely programmable via GigE Vision interface		
<b>Power consumption (aux. / 12V)</b>	2.6W	2.6W	2.5W
<b>Power consumption (PoE)</b>	3.2W	Not supported	3.2W

Table 14: Model specific specification of GC652

## Relative Response

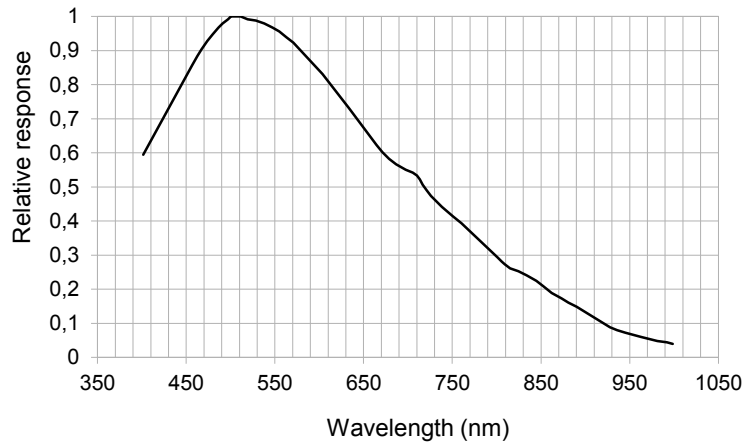


Figure 21: Relative response of GC652 Monochrome (from sensor datasheet)

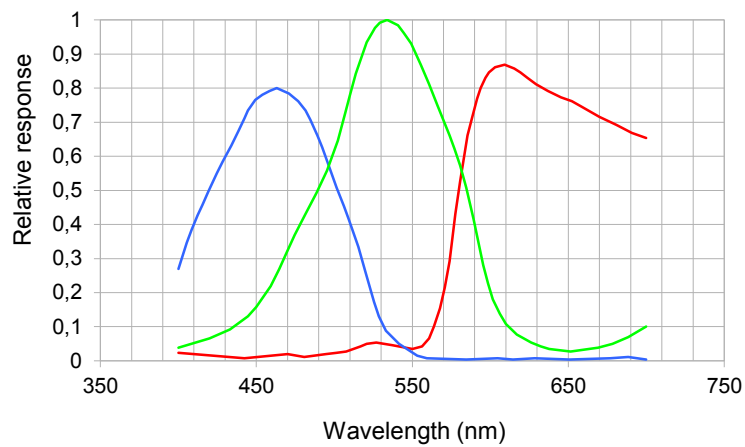


Figure 22: Relative response of GC652 Color (from sensor datasheet)

### 2.2.8 GC653M / GC653C

	GC	GC-S90	GC-BL
<b>Image Sensor</b>	Sony ICX414		
<b>Chromatics</b>	Monochrome, Color		
<b>Sensor type</b>	CCD		
<b>Sensor resolution (H x W)</b>	659x494	656x492	659x494
<b>Optical size</b>	1/2"		
<b>Pixel size (in <math>\mu\text{m}</math>)</b>	9.9 x 9.9		
<b>Analog gain (in dB)</b>	5.1 to 41.8		
<b>Shutter</b>	Progressive Scan		
<b>Exposure time</b>	10 $\mu\text{s}$ to 10s		
<b>Max. frame rate (8Bit; in Hz)</b>	97		
<b>ADC bit depth</b>	8 bit, 14 bit		
<b>Pixel data formats (mono model)</b>	Mono8, Mono16		
<b>Pixel data formats (color model)</b>	Mono8, Mono16, BayerRG8, BayerRG16		
<b>Synchronization</b>	Free run, external and software trigger (single shot, multi shot)		
<b>Exposure control</b>	Freely programmable via GigE Vision interface		
<b>Power consumption (aux. / 12V)</b>	2.6W	2.6W	2.5W
<b>Power consumption (PoE)</b>	3.2W	Not supported	3.2W

Table 15: Model specific specification of GC653

## Relative Response

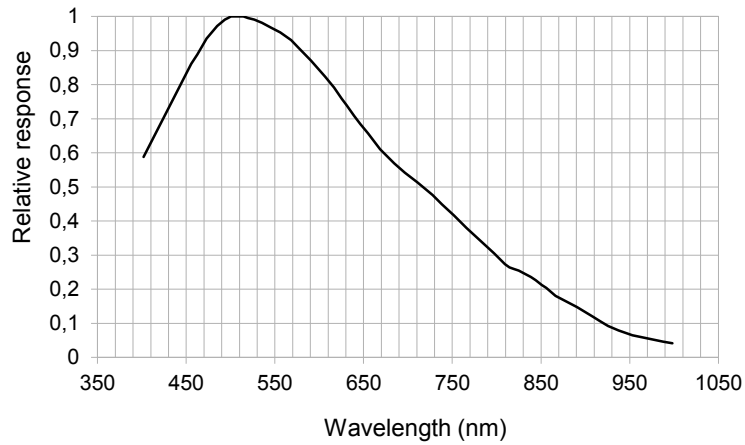


Figure 23: Relative response of GC653 Monochrome (from sensor datasheet)

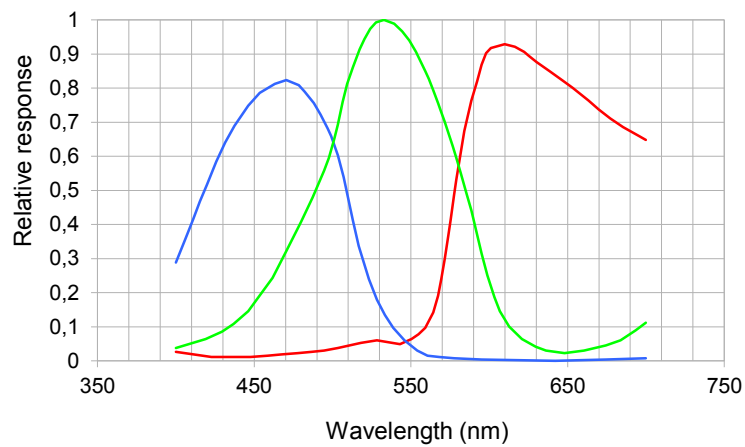


Figure 24: Relative response of GC653 Color (from sensor datasheet)

### 2.2.9 GC781M / GC781C

	GC	GC-S90	GC-BL
<b>Image Sensor</b>	Sony ICX415		
<b>Chromatics</b>	Monochrome, Color		
<b>Sensor type</b>	CCD		
<b>Sensor resolution (H x W)</b>	782x582	776x580	782x582
<b>Optical size</b>	1/2"		
<b>Pixel size (in <math>\mu\text{m}</math>)</b>	8.3 x 8.3		
<b>Analog gain (in dB)</b>	5.1 to 41.8		
<b>Shutter</b>	Progressive Scan		
<b>Exposure time</b>	10 $\mu\text{s}$ to 10s		
<b>Max. frame rate (8Bit; in Hz)</b>	68		
<b>ADC bit depth</b>	8 bit, 14 bit		
<b>Pixel data formats (mono model)</b>	Mono8, Mono16		
<b>Pixel data formats (color model)</b>	Mono8, Mono16, BayerRG8, BayerRG16		
<b>Synchronization</b>	Free run, external and software trigger (single shot, multi shot)		
<b>Exposure control</b>	Freely programmable via GigE Vision interface		
<b>Power consumption (aux. / 12V)</b>	2.6W	2.6W	2.5W
<b>Power consumption (PoE)</b>	3.2W	Not supported	3.2W

Table 16: Model specific specification of GC781

## Relative Response

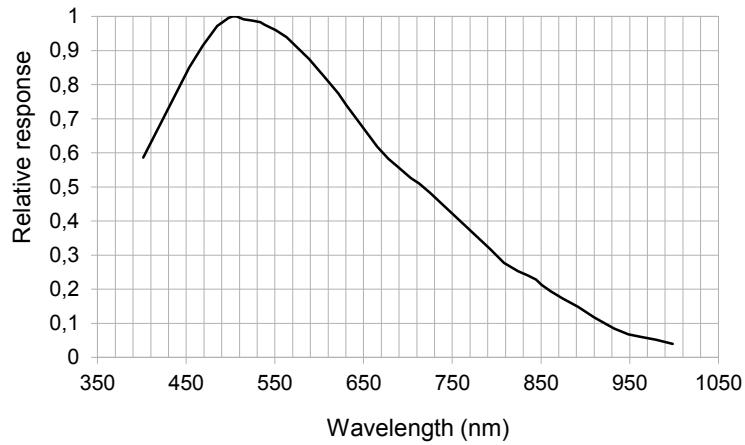


Figure 25: Relative response of GC781 Monochrome (from sensor datasheet)

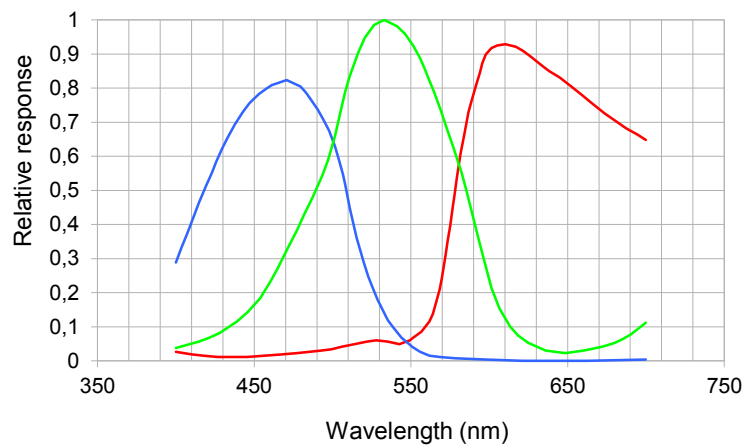


Figure 26: Relative response of GC781 Color (from sensor datasheet)

### 2.2.10 GC1031M / GC1031C

	GC	GC-S90	GC-BL
<b>Image Sensor</b>	Sony ICX204		
<b>Chromatics</b>	Monochrome, Color		
<b>Sensor type</b>	CCD		
<b>Sensor resolution (H x W)</b>	1034x779	1032x778	1034x779
<b>Optical size</b>	1/3"		
<b>Pixel size (in <math>\mu\text{m}</math>)</b>	4.65 x 4.65		
<b>Analog gain (in dB)</b>	5.1 to 41.8		
<b>Shutter</b>	Progressive Scan		
<b>Exposure time</b>	10 $\mu\text{s}$ to 10s		
<b>Max. frame rate (8Bit; in Hz)</b>	30		
<b>ADC bit depth</b>	8 bit, 14 bit		
<b>Pixel data formats (mono model)</b>	Mono8, Mono16		
<b>Pixel data formats (color model)</b>	Mono8, Mono16, BayerRG8, BayerRG16		
<b>Synchronization</b>	Free run, external and software trigger (single shot, multi shot)		
<b>Exposure control</b>	Freely programmable via GigE Vision interface		
<b>Power consumption (aux. / 12V)</b>	2.2W	2.2W	2.2W
<b>Power consumption (PoE)</b>	3.0W	Not supported	3.0W

Table 17: Model specific specification of GC1031

## Relative Response

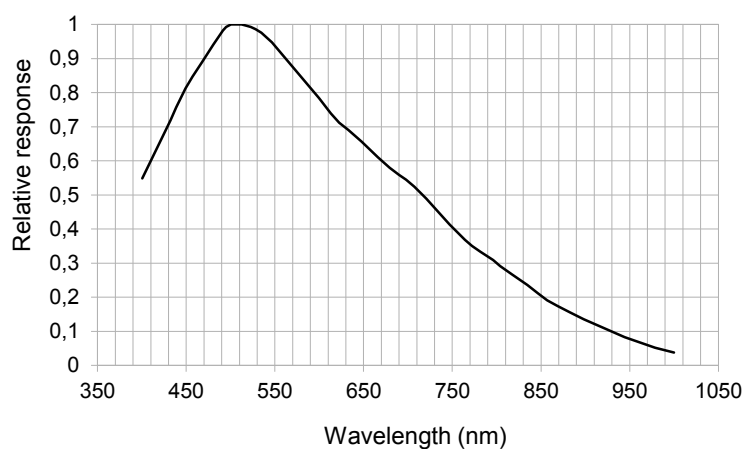


Figure 27: Relative response of GC1031 Monochrome (from sensor datasheet)

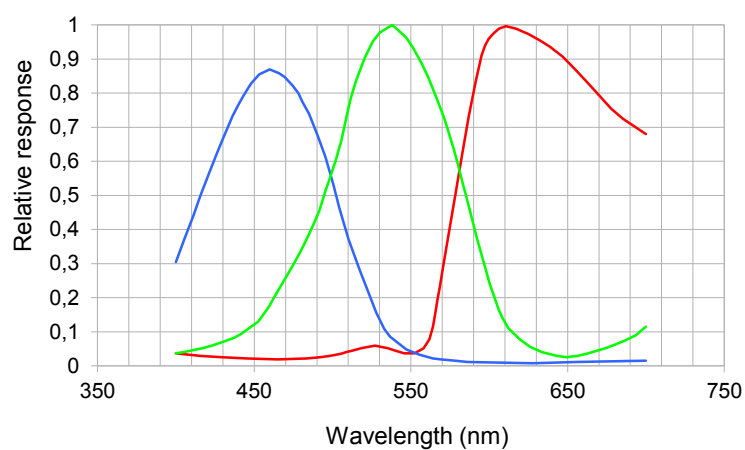


Figure 28: Relative response of GC1031 Color (from sensor datasheet)



### 2.2.11 GC1291M / GC1291C

	GC	GC-S90	GC-BL
<b>Image Sensor</b>	Sony ICX445		
<b>Chromatics</b>	Monochrome, Color		
<b>Sensor type</b>	CCD		
<b>Sensor resolution (H x W)</b>	1296x966	1288x964	1296x966
<b>Optical size</b>	1/3"		
<b>Pixel size (in <math>\mu\text{m}</math>)</b>	3.75 x 3.75		
<b>Analog gain (in dB)</b>	5.1 to 41.8		
<b>Shutter</b>	Progressive Scan		
<b>Exposure time</b>	10 $\mu\text{s}$ to 10s		
<b>Max. frame rate (8Bit; in Hz)</b>	30		
<b>ADC bit depth</b>	8 bit, 14 bit		
<b>Pixel data formats (mono model)</b>	Mono8, Mono16		
<b>Pixel data formats (color model)</b>	Mono8, Mono16, BayerRG8, BayerRG16		
<b>Synchronization</b>	Free run, external and software trigger (single shot, multi shot)		
<b>Exposure control</b>	Freely programmable via GigE Vision interface		
<b>Power consumption (aux. / 12V)</b>	2.5W	2.5W	2.5W
<b>Power consumption (PoE)</b>	3.2W	Not supported	3.2W

Table 18: Model specific specification of GC1291

## Relative Response

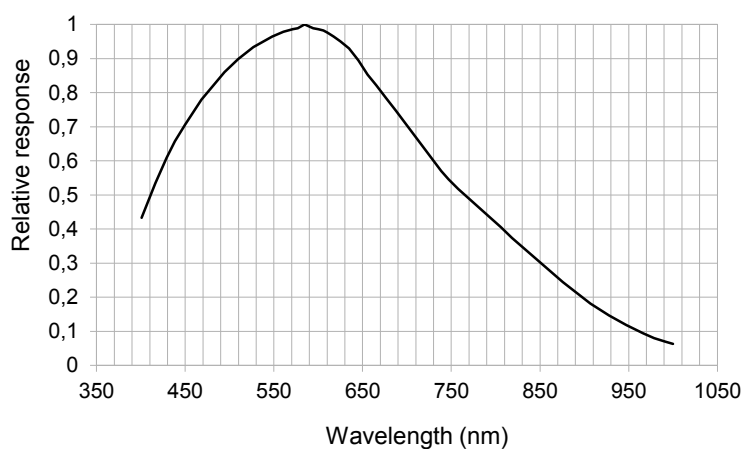


Figure 29: Relative response of GC1291 Monochrome (from sensor datasheet)

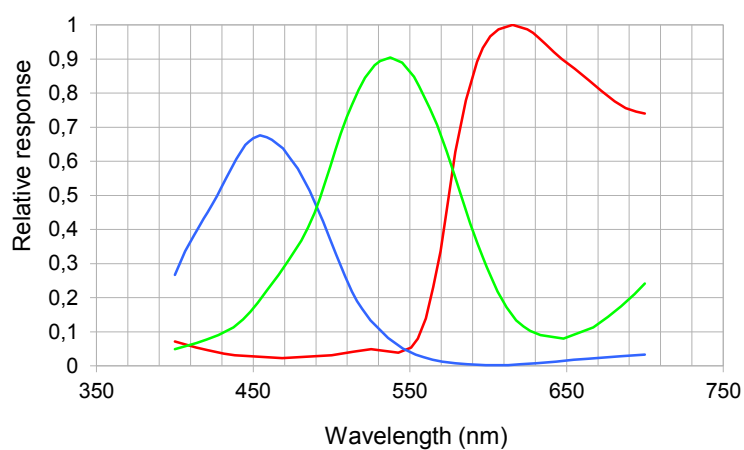


Figure 30: Relative response of GC1291 Color (from sensor datasheet)

## 2.2.12 GC1391M / GC1391C

	GC	GC-S90	GC-BL
<b>Image Sensor</b>	Sony ICX267		
<b>Chromatics</b>	Monochrome, Color		
<b>Sensor type</b>	CCD		
<b>Sensor resolution (H x W)</b>	1392x1040	1384x1038	1392x1040
<b>Optical size</b>	1/2"		
<b>Pixel size (in <math>\mu\text{m}</math>)</b>	4.65 x 4.65		
<b>Analog gain (in dB)</b>	5.1 to 41.8		
<b>Shutter</b>	Progressive Scan		
<b>Exposure time</b>	10 $\mu\text{s}$ to 10s		
<b>Max. frame rate (8Bit; in Hz)</b>	20		
<b>ADC bit depth</b>	8 bit, 14 bit		
<b>Pixel data formats (mono model)</b>	Mono8, Mono16		
<b>Pixel data formats (color model)</b>	Mono8, Mono16, BayerRG8, BayerRG16		
<b>Synchronization</b>	Free run, external and software trigger (single shot, multi shot)		
<b>Exposure control</b>	Freely programmable via GigE Vision interface		
<b>Power consumption (aux. / 12V)</b>	2.5W	2.5W	2.5W
<b>Power consumption (PoE)</b>	3.2W	Not supported	3.2W

Table 19: Model specific specification of GC1391

## Relative Response

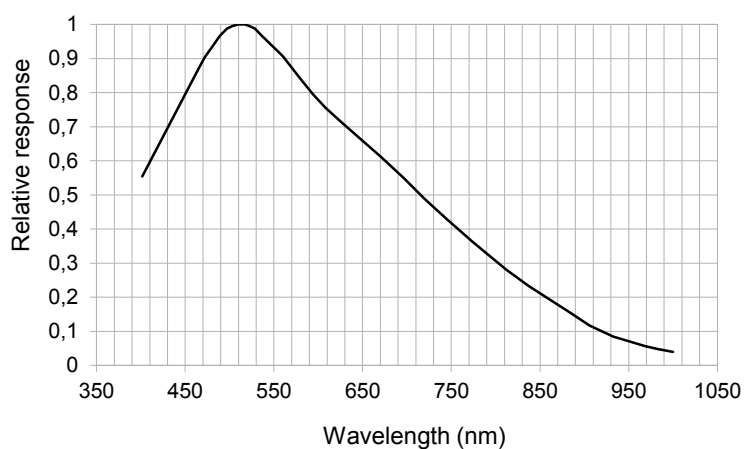


Figure 31: Relative response of GC1391 Monochrome (from sensor datasheet)

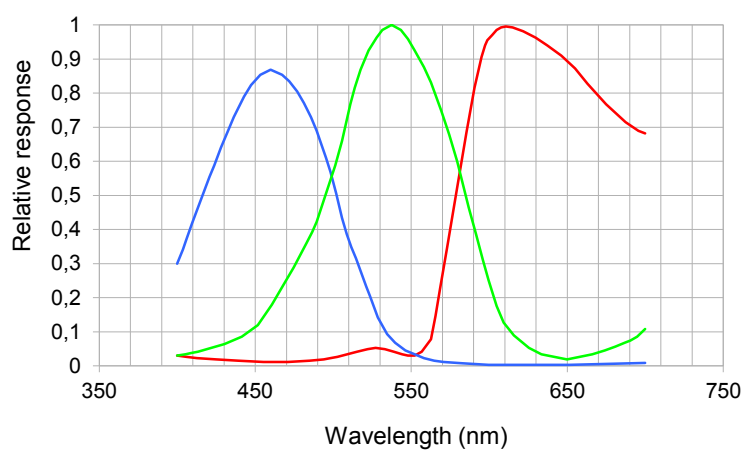


Figure 32: Relative response of GC1391 Color (from sensor datasheet)

### 2.2.13 GC1392M / GC1392C

	GC	GC-S90	GC-BL
<b>Image Sensor</b>	Sony ICX285		
<b>Chromatics</b>	Monochrome, Color		
<b>Sensor type</b>	CCD		
<b>Sensor resolution (H x W)</b>	1392x1040	1384x1038	1392x1040
<b>Optical size</b>	2/3"		
<b>Pixel size (in <math>\mu\text{m}</math>)</b>	6.45 x 6.45		
<b>Analog gain (in dB)</b>	5.1 to 41.8		
<b>Shutter</b>	Progressive Scan		
<b>Exposure time</b>	10 $\mu\text{s}$ to 10s		
<b>Max. frame rate (8Bit; in Hz)</b>	32		
<b>ADC bit depth</b>	8 bit, 14 bit		
<b>Pixel data formats (mono model)</b>	Mono8, Mono16		
<b>Pixel data formats (color model)</b>	Mono8, Mono16, BayerRG8, BayerRG16		
<b>Synchronization</b>	Free run, external and software trigger (single shot, multi shot)		
<b>Exposure control</b>	Freely programmable via GigE Vision interface		
<b>Power consumption (aux. / 12V)</b>	2.8W	2.8W	2.8W
<b>Power consumption (PoE)</b>	3.5W	Not supported	3.5W

Table 20: Model specific specification of GC1392

## Relative Response

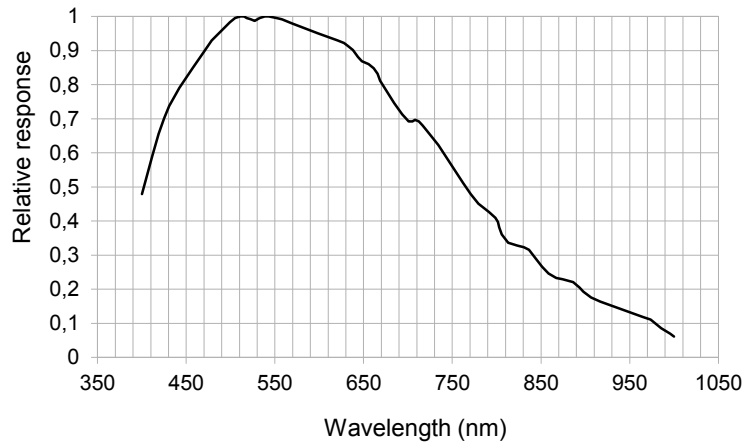


Figure 33: Relative response of GC1392 Monochrome (from sensor datasheet)

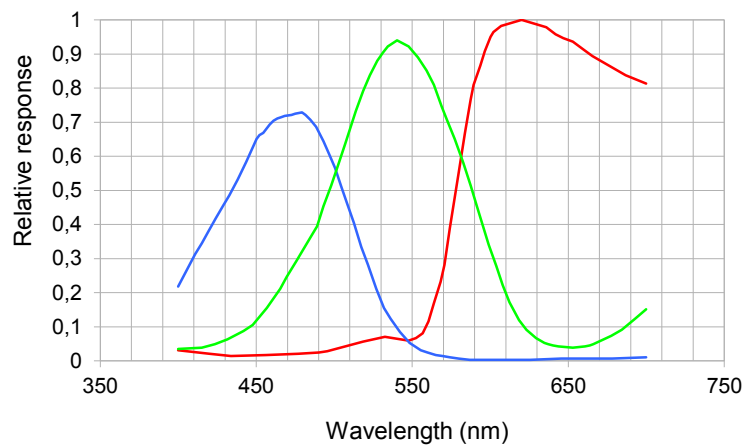


Figure 34: Relative response of GC1392 Color (from sensor datasheet)

### 2.2.14 GC1621M / GC1621C

	GC	GC-S90	GC-BL
<b>Image Sensor</b>	Sony ICX274		
<b>Chromatics</b>	Monochrome, Color		
<b>Sensor type</b>	CCD		
<b>Sensor resolution (H x W)</b>	1628x1236	1624x1234	1628x1236
<b>Optical size</b>	1/1.8"		
<b>Pixel size (in <math>\mu\text{m}</math>)</b>	4.4 x 4.4		
<b>Analog gain (in dB)</b>	5.1 to 41.8		
<b>Shutter</b>	Progressive Scan		
<b>Exposure time</b>	10 $\mu\text{s}$ to 10s		
<b>Max. frame rate (8Bit; in Hz)</b>	25		
<b>ADC bit depth</b>	8 bit, 14 bit		
<b>Pixel data formats (mono model)</b>	Mono8, Mono16		
<b>Pixel data formats (color model)</b>	Mono8, Mono16, BayerRG8, BayerRG16		
<b>Synchronization</b>	Free run, external and software trigger (single shot, multi shot)		
<b>Exposure control</b>	Freely programmable via GigE Vision interface		
<b>Power consumption (aux. / 12V)</b>	2.7W	2.7W	2.7W
<b>Power consumption (PoE)</b>	3.4W	Not supported	3.4W

Table 21: Model specific specification of GC1621

## Relative Response

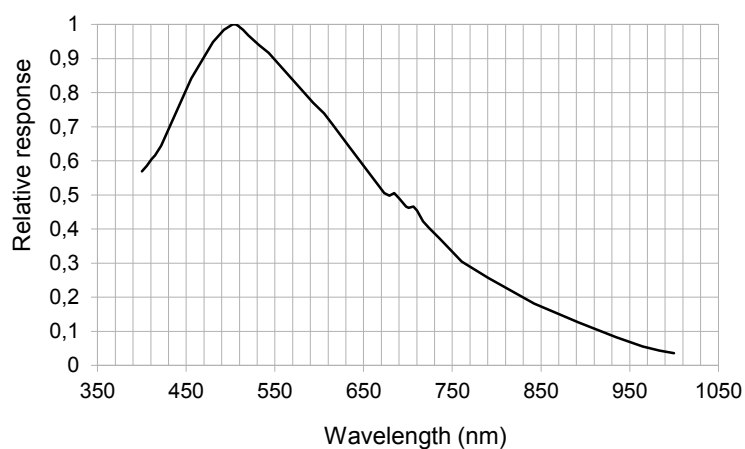


Figure 35: Relative response of GC1621 Monochrome (from sensor datasheet)

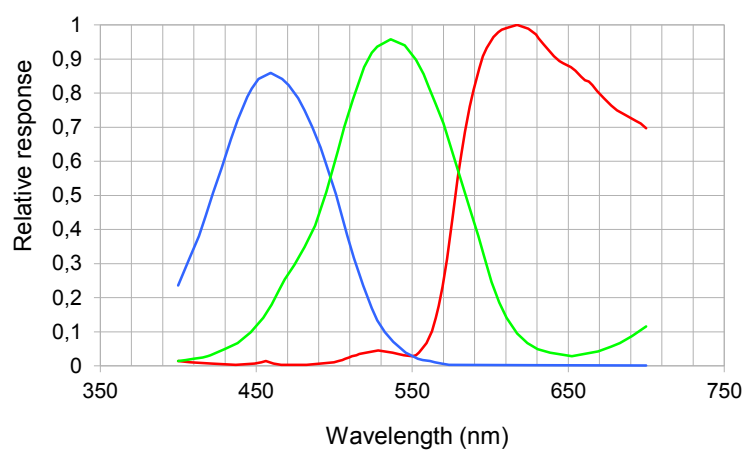


Figure 36: Relative response of GC1621 Color (from sensor datasheet)



## 2.2.15 GC2441M / GC2441C

	GC	GC-S90	GC-BL
Image Sensor	Sony ICX625		
Chromatics	Monochrome, Color		
Sensor type	CCD		
Sensor resolution (H x W)	2448x2058	2448x2056	2448x2058
Optical size	2/3"		
Pixel size (in $\mu\text{m}$ )	3.45 x 3.45		
Analog gain (in dB)	5.1 to 41.8		
Shutter	Progressive Scan		
Exposure time	10 $\mu\text{s}$ to 10s		
Max. frame rate (8Bit; in Hz)	15		
ADC bit depth	8 bit, 14 bit		
Pixel data formats (mono model)	Mono8, Mono16		
Pixel data formats (color model)	Mono8, Mono16, BayerRG8, BayerRG16		
Synchronization	Free run, external and software trigger (single shot, multi shot)		
Exposure control	Freely programmable via GigE Vision interface		
Power consumption (aux. / 12V)	3.6W	3.6W	3.6W
Power consumption (PoE)	Not supported	Not supported	4.5W

Table 22: Model specific specification of GC2441

## Relative Response

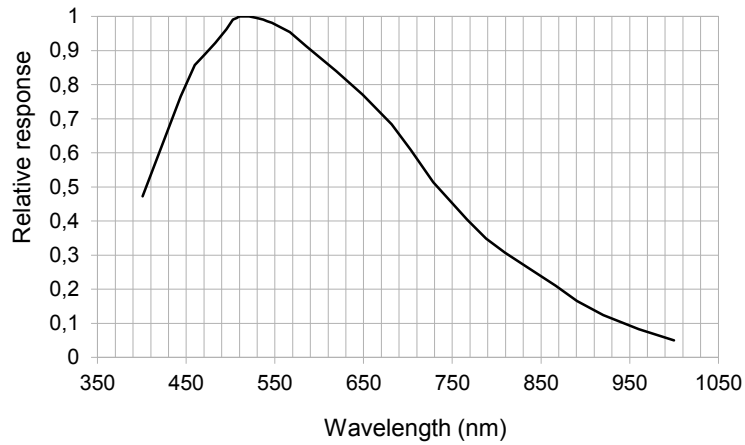


Figure 37: Relative response of GC2441 Monochrome (from sensor datasheet)

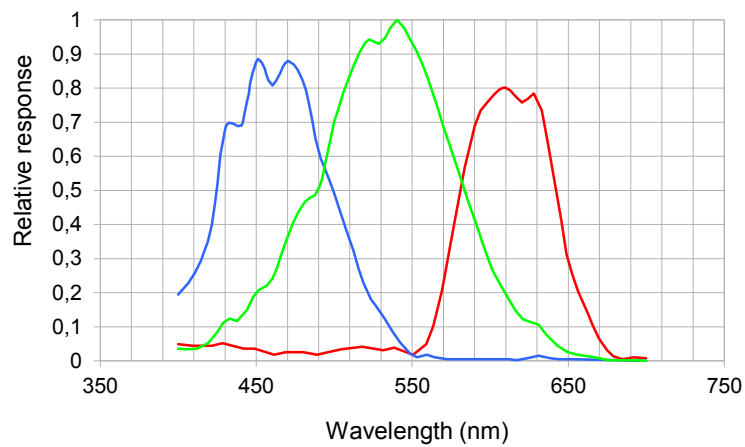


Figure 38: Relative response of GC2441 Color (from sensor datasheet)

### 2.2.16 GC1021M / GC1021C

	GC	GC-S90	GC-BL
<b>Image Sensor</b>	Truesense Imaging KAI-01050		
<b>Chromatics</b>	Monochrome, Color		
<b>Sensor type</b>	2 Tap CCD		
<b>Sensor resolution (H x W)</b>	1024 x 1024		
<b>Optical size</b>	1/2"		
<b>Pixel size (in <math>\mu\text{m}</math>)</b>	5.5 x 5.5		
<b>Analog gain (in dB)</b>	5.1 to 41.8		
<b>Shutter</b>	Progressive Scan		
<b>Exposure time</b>	10 $\mu\text{s}$ to 10s		
<b>Max. frame rate (8Bit; in Hz)</b>	61		
<b>ADC bit depth</b>	8 bit, 14 bit		
<b>Pixel data formats (mono model)</b>	Mono8, Mono16		
<b>Pixel data formats (color model)</b>	Mono8, Mono16, BayerGR8, BayerGR16		
<b>Synchronization</b>	Free run, external and software trigger (single shot, multi shot)		
<b>Exposure control</b>	Freely programmable via GigE Vision interface		
<b>Power consumption (aux. / 12V)</b>	3.4W	3.4W	3.4W
<b>Power consumption (PoE)</b>	Not supported	Not supported	4.1W

Table 23: Model specific specification of GC1021

## Relative Response

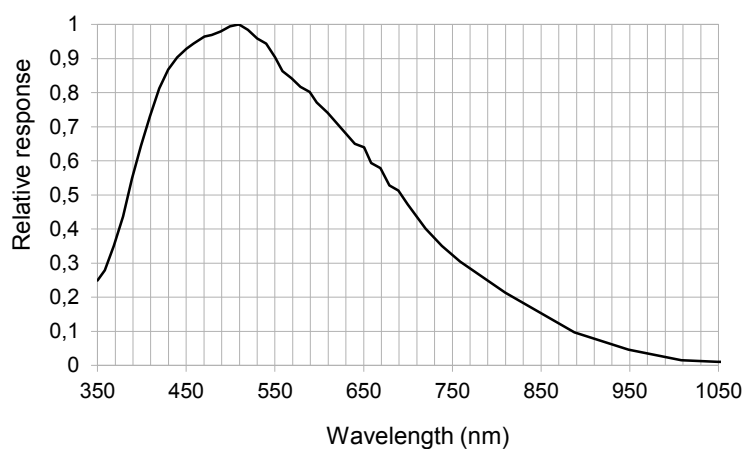


Figure 39: Relative response of GC1021 Monochrome (from sensor datasheet)

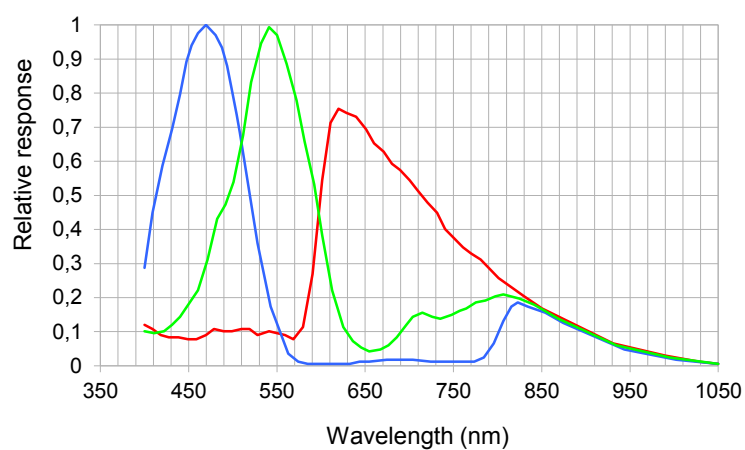


Figure 40: Relative response of GC1021 Color (from sensor datasheet)

### 2.2.17 GC1601M / GC1601C

	GC	GC-S90	GC-BL
<b>Image Sensor</b>	Truesense Imaging KAI-02050		
<b>Chromatics</b>	Monochrome, Color		
<b>Sensor type</b>	2 Tap CCD		
<b>Sensor resolution (H x W)</b>	1600 x 1200		
<b>Optical size</b>	2/3"		
<b>Pixel size (in <math>\mu\text{m}</math>)</b>	5.5 x 5.5		
<b>Analog gain (in dB)</b>	5.1 to 41.8		
<b>Shutter</b>	Progressive Scan		
<b>Exposure time</b>	10 $\mu\text{s}$ to 10s		
<b>Max. frame rate (8Bit; in Hz)</b>	61		
<b>ADC bit depth</b>	8 bit, 14 bit		
<b>Pixel data formats (mono model)</b>	Mono8, Mono16		
<b>Pixel data formats (color model)</b>	Mono8, Mono16, BayerGR8, BayerGR16		
<b>Synchronization</b>	Free run, external and software trigger (single shot, multi shot)		
<b>Exposure control</b>	Freely programmable via GigE Vision interface		
<b>Power consumption (aux. / 12V)</b>	3.5W	3.5W	3.5W
<b>Power consumption (PoE)</b>	Not supported	Not supported	4.2W

Table 24: Model specific specification of GC1601

## Relative Response

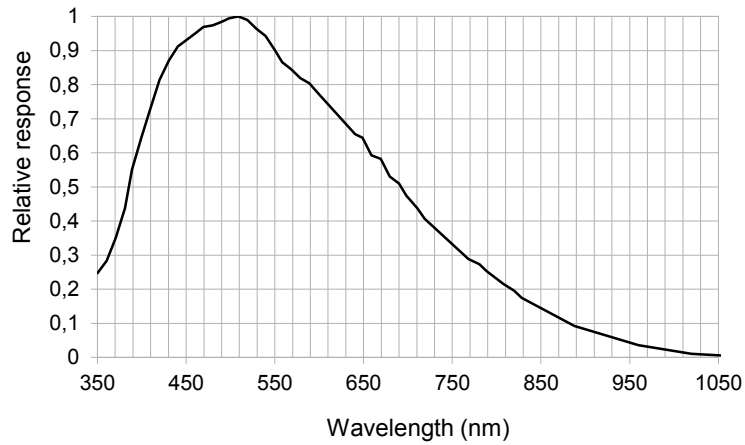


Figure 41: Relative response of GC1601 Monochrome (from sensor datasheet)

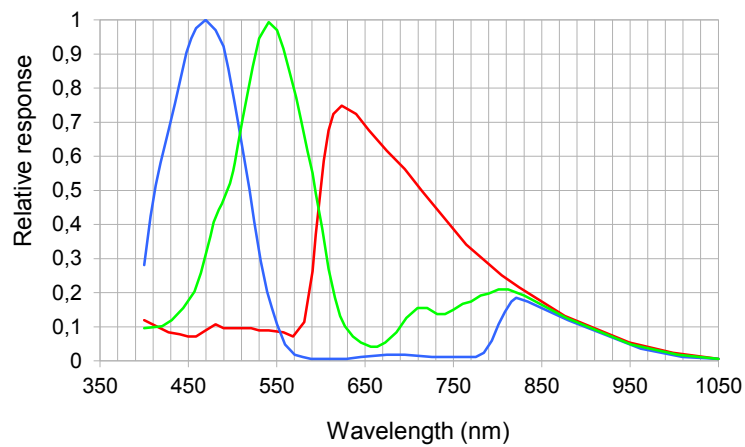


Figure 42: Relative response of GC1601 Color (from sensor datasheet)

## 2.2.18 GC1921M / GC1921C

	GC	GC-S90	GC-BL
<b>Image Sensor</b>	Truesense Imaging KAI-02150		
<b>Chromatics</b>	Monochrome, Color		
<b>Sensor type</b>	2 Tap CCD		
<b>Sensor resolution (H x W)</b>	1920 x 1080		
<b>Optical size</b>	2/3"		
<b>Pixel size (in <math>\mu\text{m}</math>)</b>	5.5 x 5.5		
<b>Analog gain (in dB)</b>	5.1 to 41.8		
<b>Shutter</b>	Progressive Scan		
<b>Exposure time</b>	10 $\mu\text{s}$ to 10s		
<b>Max. frame rate (8Bit; in Hz)</b>	33		
<b>ADC bit depth</b>	8 bit, 14 bit		
<b>Pixel data formats (mono model)</b>	Mono8, Mono16		
<b>Pixel data formats (color model)</b>	Mono8, Mono16, BayerGR8, BayerGR16		
<b>Synchronization</b>	Free run, external and software trigger (single shot, multi shot)		
<b>Exposure control</b>	Freely programmable via GigE Vision interface		
<b>Power consumption (aux. / 12V)</b>	3.6W	3.6W	3.6W
<b>Power consumption (PoE)</b>	Not supported	Not supported	4.3W

Table 25: Model specific specification of GC1921

## Relative Response

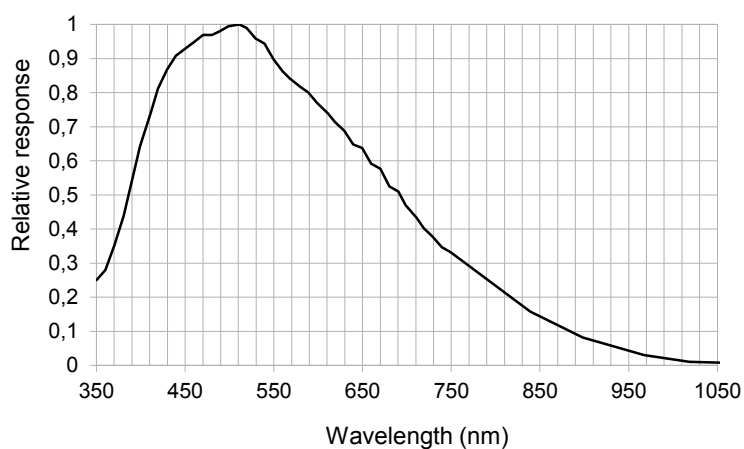


Figure 43: Relative response of GC1921 Monochrome (from sensor datasheet)

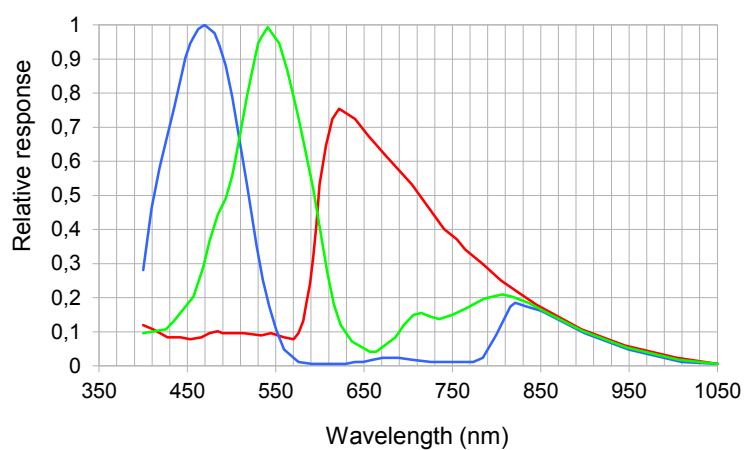


Figure 44: Relative response of GC1921 Color (from sensor datasheet)



## 2.2.19 GCP1931M / GCP1931C

	<b>GCP</b>
<b>Image Sensor</b>	Sony IMX174
<b>Chromatics</b>	Monochrome, Color
<b>Sensor type</b>	CMOS
<b>Sensor resolution (H x W)</b>	1936 x 1216
<b>Optical size</b>	1/1.2"
<b>Pixel size (in <math>\mu\text{m}</math>)</b>	5.86 x 5.86
<b>Analog gain (in dB)</b>	0 to 24
<b>Shutter</b>	Global
<b>Exposure time</b>	26 $\mu\text{s}$ to 10s
<b>Max. frame rate (8-/16Bit; in Hz)</b>	51 / 25
<b>ADC bit depth</b>	8 bit, 12 bit
<b>Pixel data formats (mono model)</b>	Mono8, Mono10Packed, Mono12Packed, Mono16
<b>Pixel data formats (color model)</b>	Mono8, Mono16, BayerRG8, BayerRG16
<b>Synchronization</b>	Free run, external and software trigger (single shot, multi shot)
<b>Exposure control</b>	Freely programmable via GigE Vision interface
<b>Power consumption (aux. / 12V)</b>	3.1W
<b>Power consumption (PoE)</b>	3.8W

Table 26: Model specific specification of GCP1931

## Relative Response

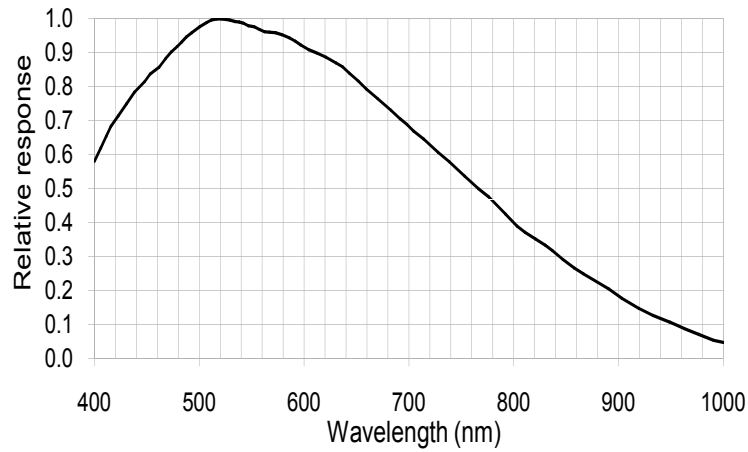


Figure 45: Relative response of GCP1931 Monochrome (from sensor datasheet)

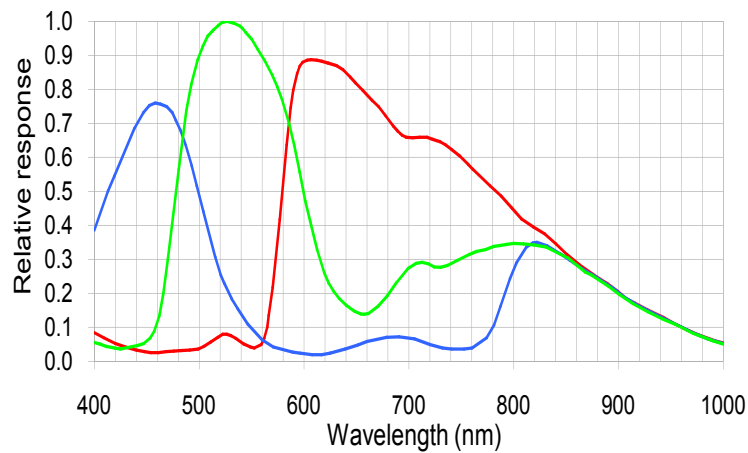


Figure 46: Relative response of GCP1931 Color (from sensor datasheet)

## 2.2.20 GCP2061M / GCP2061C

	<b>GCP</b>
<b>Image Sensor</b>	Sony IMX252
<b>Chromatics</b>	Monochrome, Color
<b>Sensor type</b>	CMOS
<b>Sensor resolution (H x W)</b>	2064 x 1544
<b>Optical size</b>	1/1.8"
<b>Pixel size (in <math>\mu\text{m}</math>)</b>	3.45 x 3.45
<b>Analog gain (in dB)</b>	0 to 24
<b>Shutter</b>	Global Shutter
<b>Exposure time</b>	26 $\mu\text{s}$ to 10s
<b>Max. frame rate (8-/16Bit; in Hz)</b>	38 / 19
<b>ADC bit depth</b>	8 bit, 12 bit
<b>Pixel data formats (mono model)</b>	Mono8, Mono10Packed, Mono12Packed, Mono16
<b>Pixel data formats (color model)</b>	Mono8, Mono16, BayerRG8, BayerRG16
<b>Synchronization</b>	Free run, external and software trigger (single shot, multi shot)
<b>Exposure control</b>	Freely programmable via GigE Vision interface
<b>Power consumption (aux. / 12V)</b>	3.6W
<b>Power consumption (PoE)</b>	4.6W

Table 27: Model specific specification of GCP2061

## Relative Response

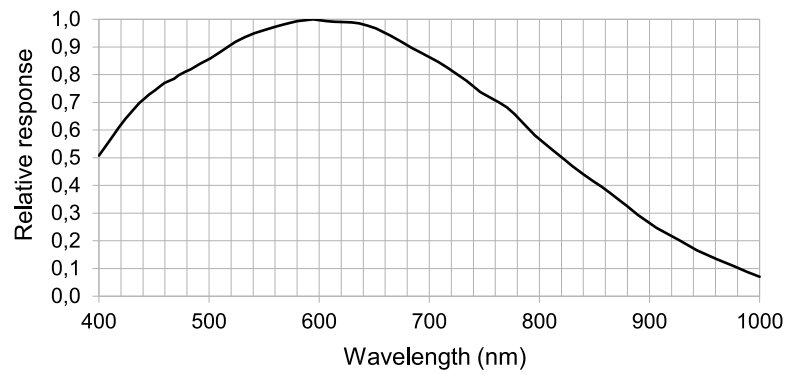


Figure 47: Relative response of GCP2061 Monochrome (from sensor datasheet)

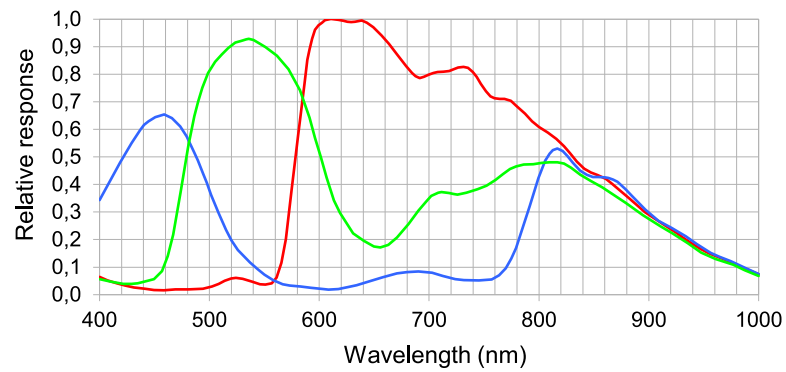


Figure 48: Relative response of GCP2061 Color (from sensor datasheet)

## 2.2.21 GCP2461M / GCP2461C

	GCP
Image Sensor	Sony IMX250
Chromatics	Monochrome, Color
Sensor type	CMOS
Sensor resolution (H x W)	2464 x 2056
Optical size	2/3"
Pixel size (in $\mu\text{m}$ )	3.45 x 3.45
Analog gain (in dB)	0 to 24
Shutter	Global Shutter
Exposure time	28 $\mu\text{s}$ to 10s
Max. frame rate (8-/16Bit; in Hz)	24 / 12
ADC bit depth	8 bit, 12 bit
Pixel data formats (mono model)	Mono8, Mono10Packed, Mono12Packed, Mono16
Pixel data formats (color model)	Mono8, Mono16, BayerRG8, BayerRG16
Synchronization	Free run, external and software trigger (single shot, multi shot)
Exposure control	Freely programmable via GigE Vision interface
Power consumption (aux. / 12V)	3.6W
Power consumption (PoE)	4.6W

Table 28: Model specific specification of GCP2461

## Relative Response

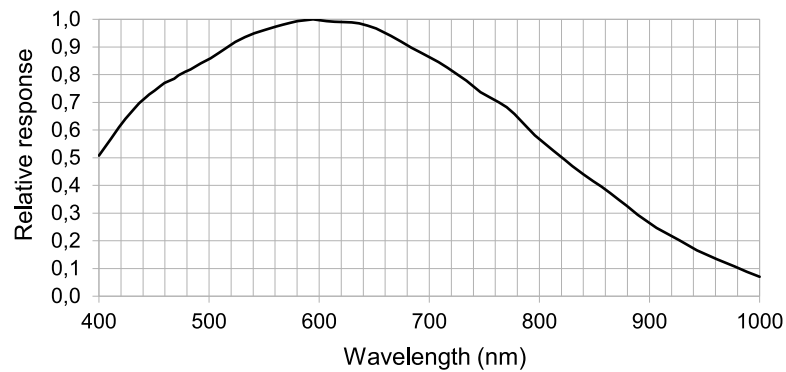


Figure 49: Relative response of GCP2461 Monochrome (from sensor datasheet)

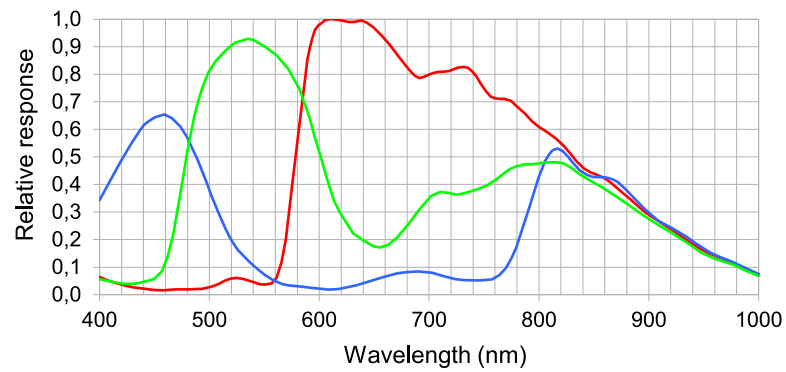


Figure 50: Relative response of GCP2461 Color (from sensor datasheet)

## 2.2.22 GCP1941M / GCP1941C

	GCP
Image Sensor	Sony ICX674
Chromatics	Monochrome, Color
Sensor type	4 Tap CCD
Sensor resolution (H x W)	1936 x 1456
Optical size	2/3"
Pixel size (in $\mu\text{m}$ )	4.54 x 4.54
Analog gain (in dB)	12 to 24
Shutter	Global Shutter
Exposure time	10 $\mu\text{s}$ to 10s
Max. frame rate (8-/16Bit; in Hz)	36 / 17
ADC bit depth	8 bit, 14 bit
Pixel data formats (mono model)	Mono8, Mono10Packed, Mono12Packed, Mono16
Pixel data formats (color model)	Mono8, Mono16, BayerRG8, BayerRG16
Synchronization	Free run, external and software trigger (single shot, multi shot)
Exposure control	Freely programmable via GigE Vision interface
Power consumption (aux. / 12V)	4.1W
Power consumption (PoE)	5.2W

Table 29: Model specific specification of GCP1941

## Relative Response

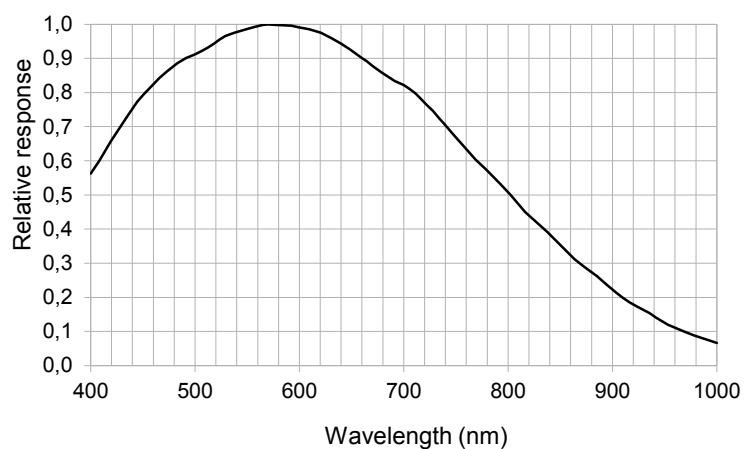


Figure 51: Relative response of GCP1941 Monochrome (from sensor datasheet)

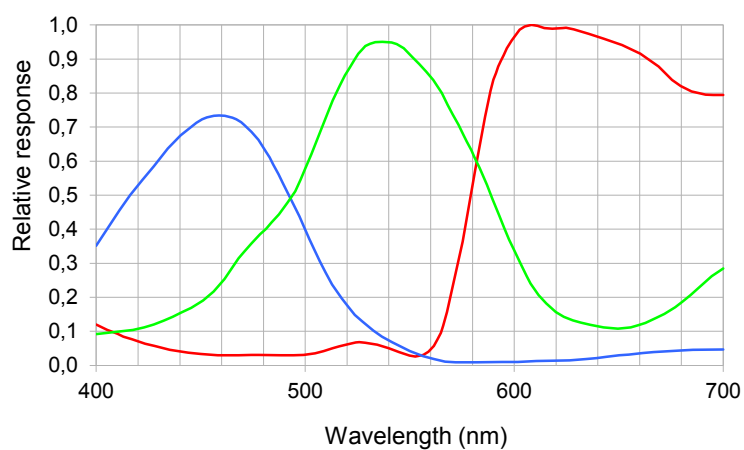


Figure 52: Relative response of GCP1941 Color (from sensor datasheet)



### 2.2.23 GCP2751M / GCP2751C

	GCP
Image Sensor	Sony ICX694
Chromatics	Monochrome, Color
Sensor type	4 Tap CCD
Sensor resolution (H x W)	2752 x 2208
Optical size	1"
Pixel size (in $\mu\text{m}$ )	4.54 x 4.54
Analog gain (in dB)	12 to 24
Shutter	Global Shutter
Exposure time	10 $\mu\text{s}$ to 10s
Max. frame rate (8-/16Bit; in Hz)	18 / 8
ADC bit depth	8 bit, 14 bit
Pixel data formats (mono model)	Mono8, Mono10Packed, Mono12Packed, Mono16
Pixel data formats (color model)	Mono8, Mono16, BayerGR8, BayerGR16
Synchronization	Free run, external and software trigger (single shot, multi shot)
Exposure control	Freely programmable via GigE Vision interface
Power consumption (aux. / 12V)	4.2W
Power consumption (PoE)	5.6W

Table 30: Model specific specification of GCP2751

## Relative Response

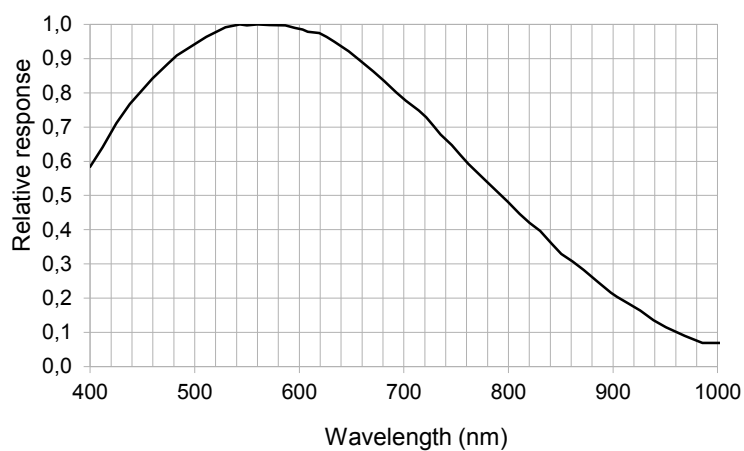


Figure 53: Relative response of GCP2751 Monochrome (from sensor datasheet)

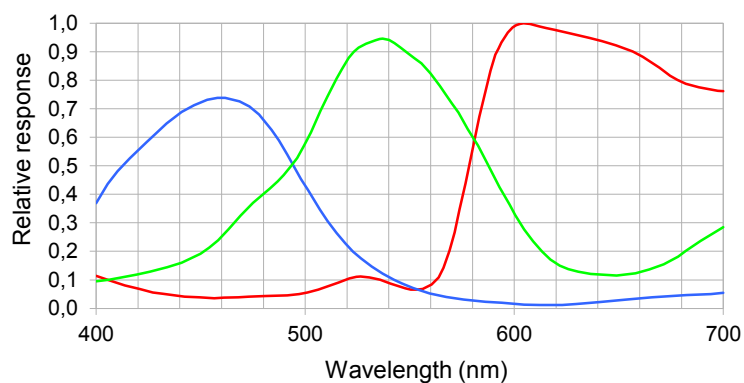


Figure 54: Relative response of GCP2751 Color (from sensor datasheet)

## 2.2.24 GCP3381M / GCP3381C

	GCP
Image Sensor	Sony ICX814
Chromatics	Monochrome, Color
Sensor type	4 Tap CCD
Sensor resolution (H x W)	3376 x 2704
Optical size	1"
Pixel size (in $\mu\text{m}$ )	3.69 x 3.69
Analog gain (in dB)	12 to 24
Shutter	Global Shutter
Exposure time	10 $\mu\text{s}$ to 10s
Max. frame rate (8-/16Bit; in Hz)	12 / 5
ADC bit depth	8 bit, 14 bit
Pixel data formats (mono model)	Mono8, Mono10Packed, Mono12Packed, Mono16
Pixel data formats (color model)	Mono8, Mono16, BayerRG8, BayerRG16
Synchronization	Free run, external and software trigger (single shot, multi shot)
Exposure control	Freely programmable via GigE Vision interface
Power consumption (aux. / 12V)	4.4W
Power consumption (PoE)	5.6W

Table 31: Model specific specification of GCP3381

## Relative Response

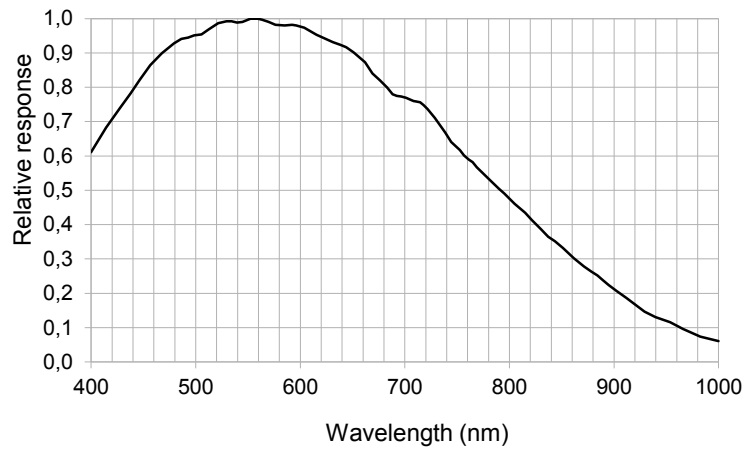


Figure 55: Relative response of GCP3381 Monochrome (from sensor datasheet)

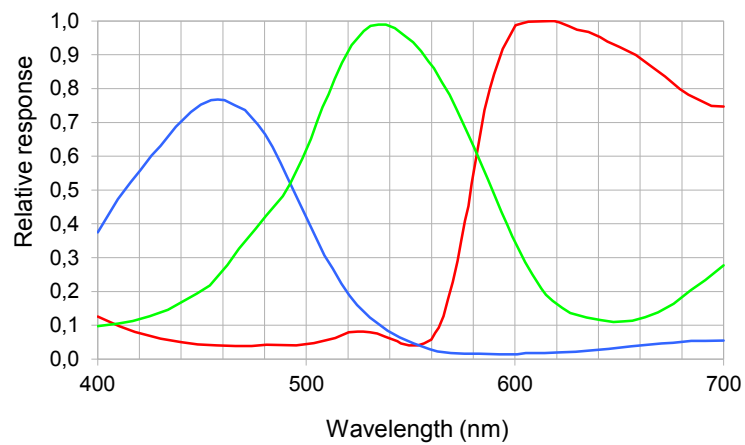


Figure 56: Relative response of GCP3381 Color (from sensor datasheet)

## 2.2.25 GCP4241M / GCP4241C

	GCP
Image Sensor	Sony ICX834
Chromatics	Monochrome, Color
Sensor type	4 Tap CCD
Sensor resolution (H x W)	4240 x 2824
Optical size	1"
Pixel size (in $\mu\text{m}$ )	3.1 x 3.1
Analog gain (in dB)	12 to 24
Shutter	Global Shutter
Exposure time	10 $\mu\text{s}$ to 10s
Max. frame rate (8-/16Bit; in Hz)	9 / 4
ADC bit depth	8 bit, 14 bit
Pixel data formats (mono model)	Mono8, Mono10Packed, Mono12Packed, Mono16
Pixel data formats (color model)	Mono8, Mono16, BayerRG8, BayerRG16
Synchronization	Free run, external and software trigger (single shot, multi shot)
Exposure control	Freely programmable via GigE Vision interface
Power consumption (aux. / 12V)	4.7W
Power consumption (PoE)	5.8W

Table 32: Model specific specification of GCP4241

## Relative Response

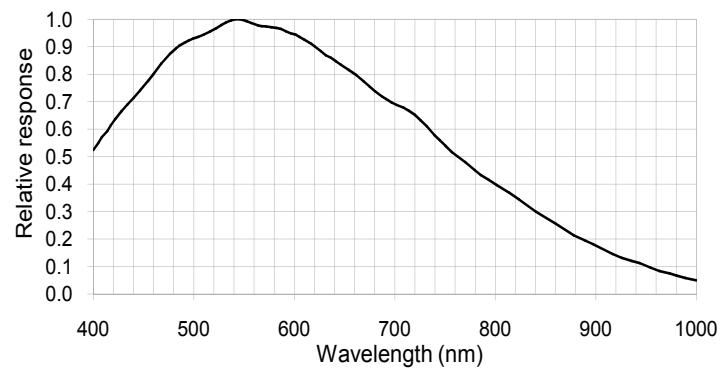


Figure 57: Relative response of GCP4241 Monochrome (from sensor datasheet)

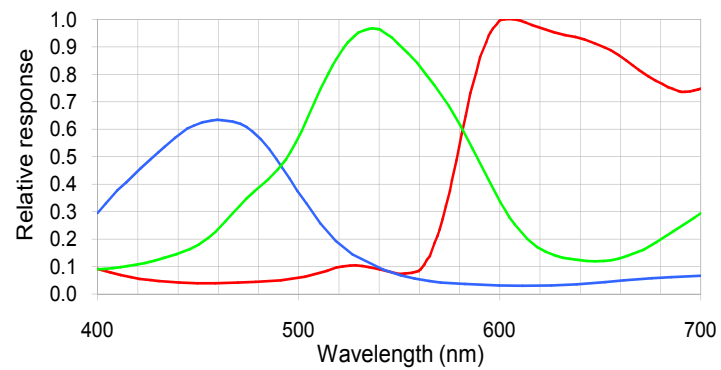


Figure 58: Relative response of GCP4241 Color (from sensor datasheet)

### 2.3 Physical Interfaces

All cameras are equipped with two physical interfaces - a circular Hirose jack providing the camera's power and digital IO lines and a RJ45 jack for 100/1000Mbit/s Ethernet communication. Figure 59 shows the general connecting scheme.

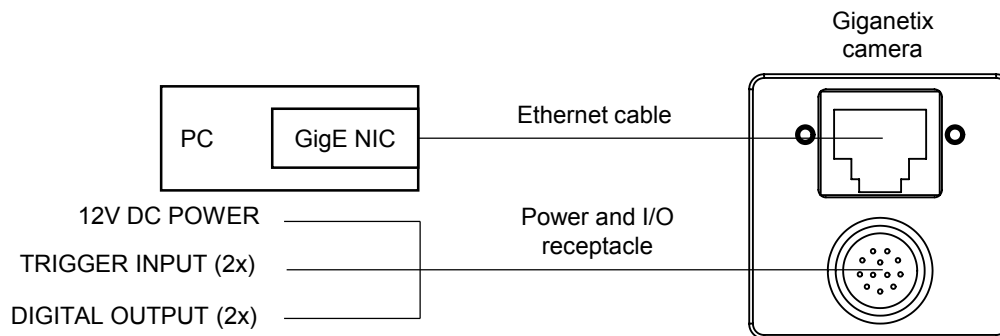


Figure 59: Connecting scheme

#### 2.3.1 Ethernet Interface

The Ethernet Interface provides configuration access to the camera and is also used for image data transmission. The connector is a standardize RJ45 jack, assigned like shown in Table 33 below.

Ethernet Connector Type		RJ45, Ethernet 1000BaseT, 802.3 compliant	
Pin no.	Signal	Description	
1	BI_DA+	Bi-directional pair +A	
2	BI_DA-	Bi-directional pair -A	
3	BI_DB+	Bi-directional pair +B	
4	BI_DC+	Bi-directional pair +C	
5	BI_DC-	Bi-directional pair -C	
6	BI_DB-	Bi-directional pair -B	
7	BI_DD+	Bi-directional pair +D	
8	BI_DD-	Bi-directional pair -D	

Table 33: Ethernet connector type and assignment

### Status LEDs

The Ethernet connector provides a yellow and a green LED. The green LED indicates the status of the Ethernet link and its activity, while the yellow LED indicates the status of the camera. The description of the different statuses of these LEDs can be found in Table 34 and Table 35 below.

Green LED (Status)	Description
Off	No link
Solid on	Link on / Ethernet link exists
Blinking	Indicates ongoing Ethernet activity

Table 34: Ethernet Green LED status

Yellow LED (Status)	Description
Off	Not powered
Solid on	Power on / Status OK
One blink, then Off	No user firmware / Factory firmware active
Two blinks, then Off	Watchdog timer timeout error
Three blinks, then Off	User firmware data CRC error
Four blinks, then Off	Internal FPGA configuration error

Table 35: Ethernet Yellow LED status

### Cabling Requirements

To connect the camera to a network, at least a straight UTP (Unshielded Twisted Pair) CAT5e cable needs to be used in environments with low or no EMI. In environments with higher EMI, a STP (Shielded Twisted Pair) CAT6 cable is recommended. The scheme for Straight-through patch cable is shown on Figure 60.

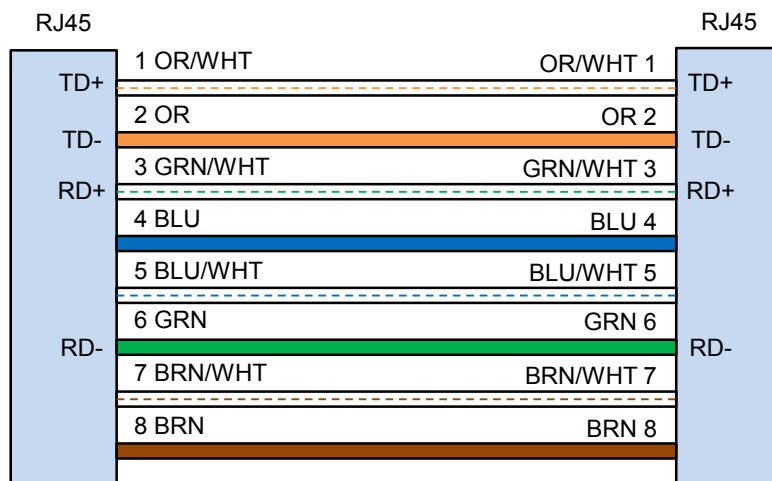


Figure 60: Straight-through cable scheme



### 2.3.2 Power and I/O-Interface

Beside the Ethernet interface for communication and data transmission, all cameras are equipped with a *Power and I/O-Interface*. Via this interface the cameras provide access to two digital input and two digital output lines, as well as their main power supply input. Depending on the type of camera there are two different kind of connector types used, shown in Table 36.

Model	Connector Type	Receptacle
<b>GC</b> (standard housing)	12-pin Circular Hirose	HR10A-10P-12S
<b>GC-S90</b> (angled 90° housing)	12-pin Circular Hirose	HR10A-10P-12S
<b>GCP</b> (standard housing)	12-pin Circular Hirose	HR10A-10P-12S
<b>GC-BL</b> (board level)	10-pin Molex Picoblade	53398-1071, 10-pins

Table 36: Power and I/O-interface connector type per model

#### 2.3.2.1 12-pin Circular Hirose Connector

The housed Giganetix standard and 90° angled cameras are equipped with a 12-pin circular Hirose receptacle to provide access to the power interface as well as the input and output lines. Figure 61 shows the pin and connector orientation on the back of the camera housing, Table 37 shows the corresponding pin assignment.

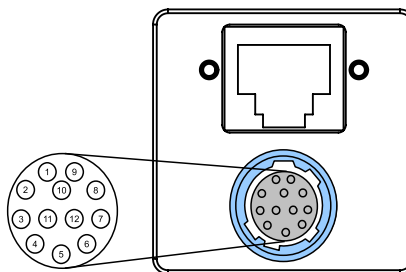


Figure 61: 12-pin circular Hirose receptacle - Pin and connector orientation

Pin no.	Signal
1	Power GND
2	DC power supply
3	Output 1 -
4	Output 1 +
5	Input 2 -
6	Input 2 +
7	Input 1 +
8	Input 1 -
9	Output 2 -
10	Output 2 +
11	Input 1 +
12	Input 1 -

Table 37: 12-pin circular Hirose receptacle - Pin assignment



### Note

The 12-pin connector on the camera is a Hirose receptacle and can be used with a HR10A-10P-12S or equivalent.



### Caution

Only cameras with PoE provide a polarity protection on the power input circuit; voltage reversal on models without PoE will damage the camera!

### Cabling Requirements

A single 12-pin Hirose receptacle is used to power the camera and provide access to its input and output lines. When assembling the 12 pin Hirose connector on one side of the cable, care must be taken to follow the pin arrangement shown in Table 37 (determining pin 1 from the supplied drawing is critical). It is recommended to use a shielded twisted pair cable to avoid EMI, the maximum length should not exceed 10m.

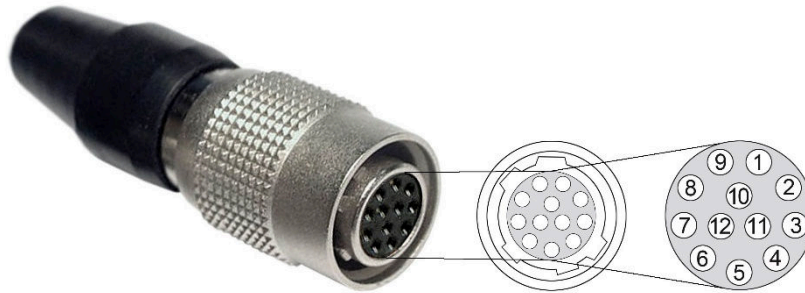


Figure 62: Hirose 12-pin plug connector



#### Note

The 12 pin connector for the cable is a Hirose plug HR10A-10P-12S(73) (or equivalent).



#### Caution

An incorrect pin alignment or connector can damage the camera.

### 2.3.2.2 10-pin Molex Picoblade Connector

The board level Gigasetix cameras (GC-BL) are equipped with a 10-pin Molex Picoblade receptacle to provide access to the power interface as well as the input and output lines. Figure 52 shows the pin and connector orientation on the back of the camera housing, Table 38 shows the corresponding pin assignment.

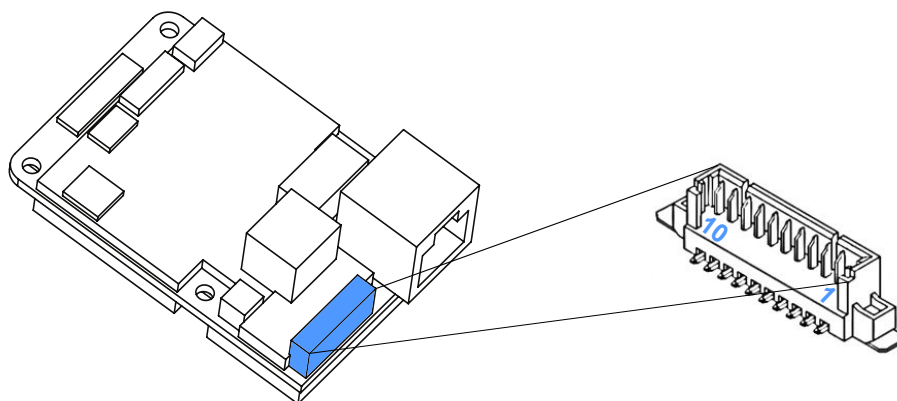


Figure 63: 10-pin Molex Picoblade receptacle - Pin and connector orientation

Pin no.	Signal
1	DC power supply
2	Power GND
3	Output 2 +
4	Output 2 -
5	Output 1 +
6	Output 1 -
7	Input 1 +
8	Input 1 -
9	Input 2 +
10	Input 2 -

Table 38: 12-pin circular Hirose receptacle - Pin assignment



#### Note

The 10-pin connector on the camera mainboard is a Molex Picoblade 53398-1071 with 10-pins.

### 2.3.2.3 Input Lines (Electrical Specifications)

All cameras are equipped with two physical input lines designated as input line 1 and input line 2. The input lines are accessed via the power and I/O interface receptacle on the back of the camera. Each input line is opto-isolated. Table 39 shows the operational limits of the trigger input lines, while Figure 64 shows their electrical scheme.

Description	Limits
Recommended operating voltage	+0 to +24 VDC
Voltage level representing logical 0	+0 to +1.4 VDC
Region where the transition threshold occurs; the logical state is not defined in this region	> +1.4 to +2.2 VDC
Voltage level representing logical 1	> +2.2 VDC
Absolute maximum; the camera may be damaged when the absolute maximum is exceeded	+30.0 VDC
The current draw for each input line	5 to 15 mA

Table 39: Electrical specification for trigger input (operational limits)

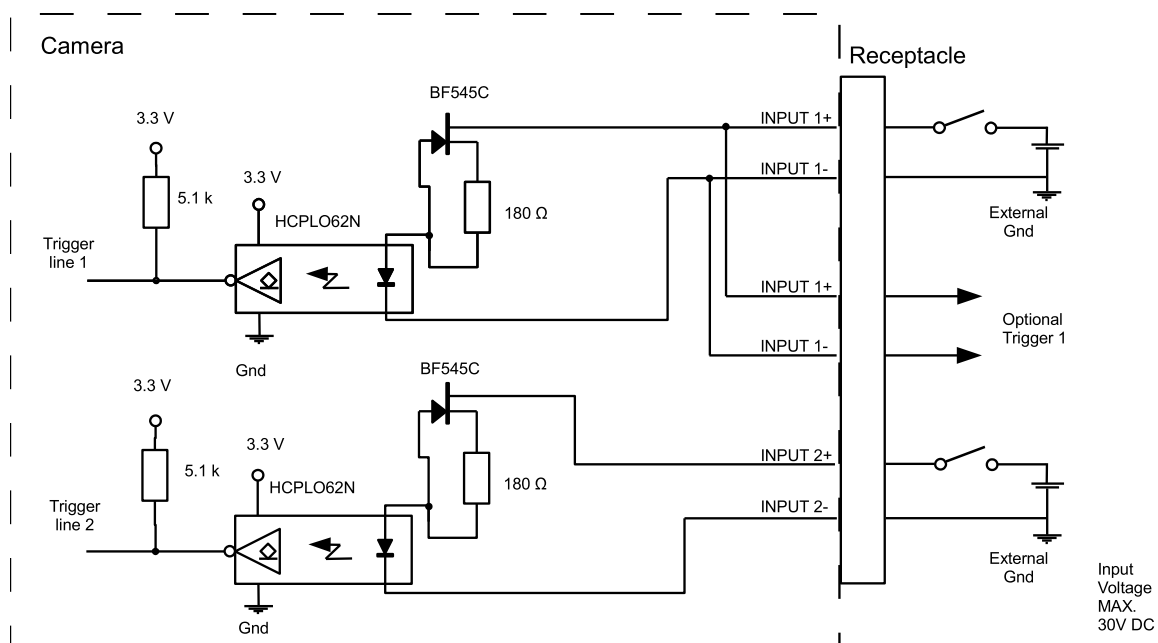


Figure 64: Trigger input scheme



### Caution

Exceeding the limits shown in Table 39 or renegeing the wiring polarity shown in Figure 64 can seriously damage the device!

### 2.3.2.4 Output Lines (Electrical Specifications)

All cameras are equipped with two physical output lines, designated as output line 1 and output line 2. The output lines are accessed via the power and I/O interface receptacle, Table 40 shows the operational limits of the trigger input lines, Figure 65 shows their electrical scheme. Each output line is opto-isolated.

Description	Limits
The I/O output may operate erratically	< +3.3 VDC
Recommended operating voltage	+3.3 to +24 VDC
Absolute maximum; the camera may be damaged if the absolute maximum is exceeded	+30.0 VDC
The maximum current surge for outputs	25 mA

Table 40: Electrical specification for digital output

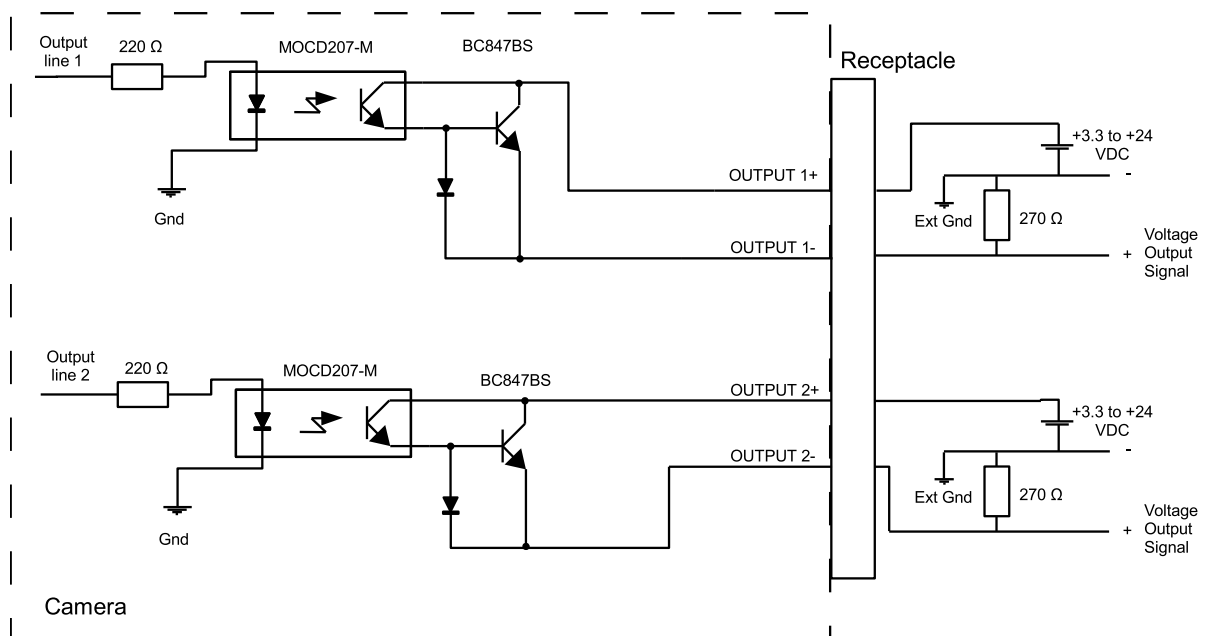


Figure 65: Digital output scheme



### Caution

Exceeding the limits shown in Table 40 or renegeing the wiring polarity shown in Figure 65 can seriously damage the device!

### 2.3.3 Temperature Specification and Heat Dissipation

The temperature specification given for storing and operation of all devices are measured at any location of the camera's housing. If the camera is delivered without a housing, the specified range refers to the direct ambient temperature of the board-set, at any location. In operation it must be ensured that the internal heat generation of the camera is dissipated sufficiently to make the device or its environment not exceed the specified borders. The camera will steadily heat up in the first hour of operation and should be monitored.

Beside the risk of damage, a too high camera temperature also decreases the image quality of the sensor significantly. The thermal noise generated in silicon based sensors raises exponentially with the temperature, hereby the useful signal falls rapidly.

As every application and environment has its own characteristics, SMARTERK Vision can only suggest general strategies to keep the camera's temperature low:

- Mount housed cameras with at least one complete side of the housing to a massive heat conductive material (e.g. aluminum); make sure that the whole surface is constantly in touch
- Active cooling of the camera by a fan will significantly decrease the temperature
- Keep the ambience temperature as low as possible

Board level cameras:

- If mounted into another closed device, make sure to offer a constant heat exchange by e.g. an air flow
- Additional active / passive heat sinking of the critical components (Sensor Head, Flash, FPGA, DDR, Ethernet Physical, PoE driver etc.) allows for higher ambient temperatures (at own risk)

#### Example

Figure 66 gives an example of the thermal behavior of a Gigasetix camera mounted to different heat conductors, described in Table 41. The used camera is a GC1921M with a 2 Tap Truesense Imaging sensor, which was chosen as one with the highest power consumption (3.6 Watts) in the Gigasetix lineup.




Color	Label	Description
	<i>Not mounted</i>	Camera is placed on a material with a very low heat conductivity (plastic); the main heat dissipation occurs over the surrounding air
	<i>Aluminum (loose)</i>	Camera is placed loose on a construction profile (150x70x30 mm) of a material with a very good heat conductivity (aluminum)
	<i>Aluminum (well mounted)</i>	Camera is well mounted on a construction profile (150x70x30 mm) of a material with a very good heat conductivity (aluminum)

Table 41: Description of the curves in Figure 66

In each setup, the camera and its heat conductor are exposed to the environment temperature of 22.5°C, until all match (1). As soon as the camera is powered (2) it starts to heat immediately (3) and reaches its maximum after around one hour (4). The difference in temperature between the sample setups is significantly; the camera which is not mounted to any heat conductor after one hour in an environmental temperature of 22.5°C have camera temperature at 50.4°C. With a small aluminum heat conductor the camera temperature drops about 12 to 15°C.

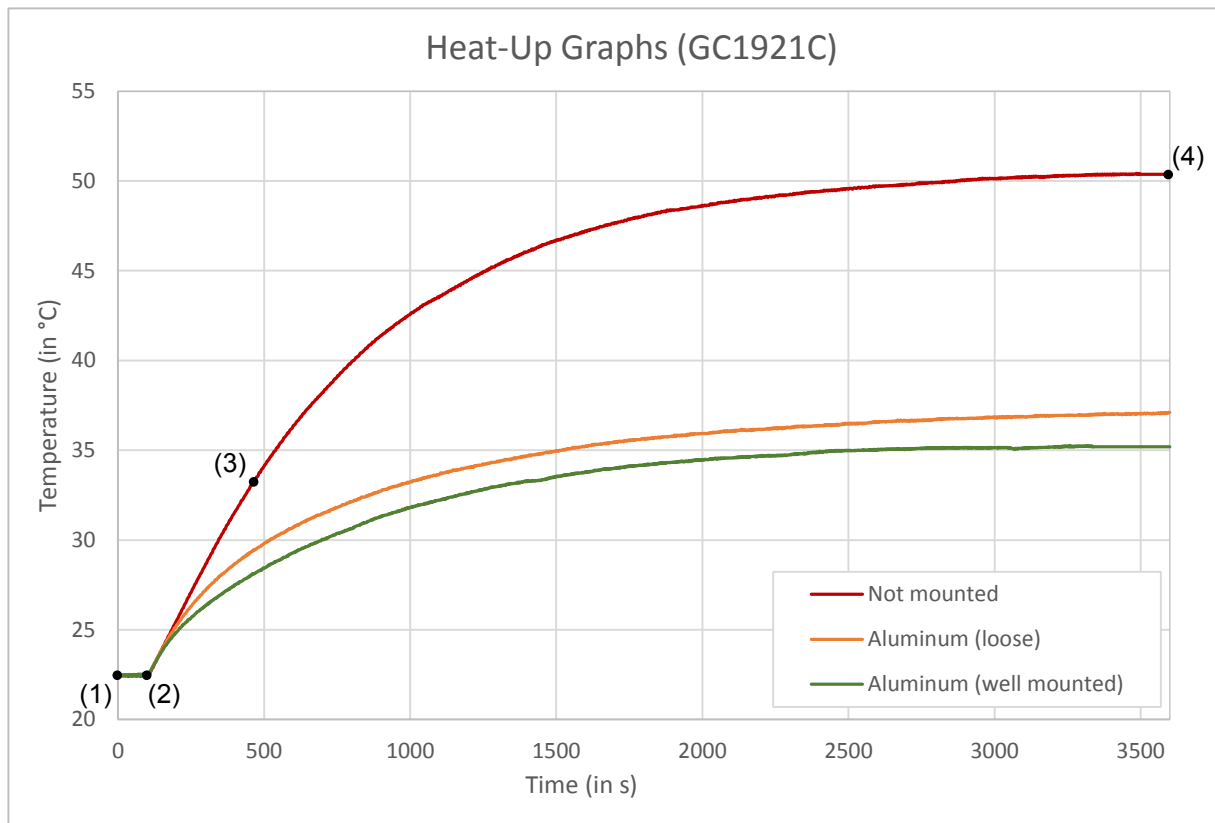


Figure 66: Example of heat up behavior of a camera with different thermal connections



### 2.3.4 IR-Cut Filter

The spectral sensitivity of the CCD/CMOS image sensors extends into the near-infrared range, what can result in for the human eye unnatural-looking images on color camera models. To allow an accurate reproduction of images from color image sensors, IR-cut filters are used.

IR-cut filters are short pass filters that block near infrared light of wavelengths longer than approximately 660nm and pass visible light. All color camera models are equipped with an IR-cut filter as standard, monochrome models do not have an IR-cut filter installed by default. Figure 67 below shows the transmission curve of the filter used in the Giganetix camera family.

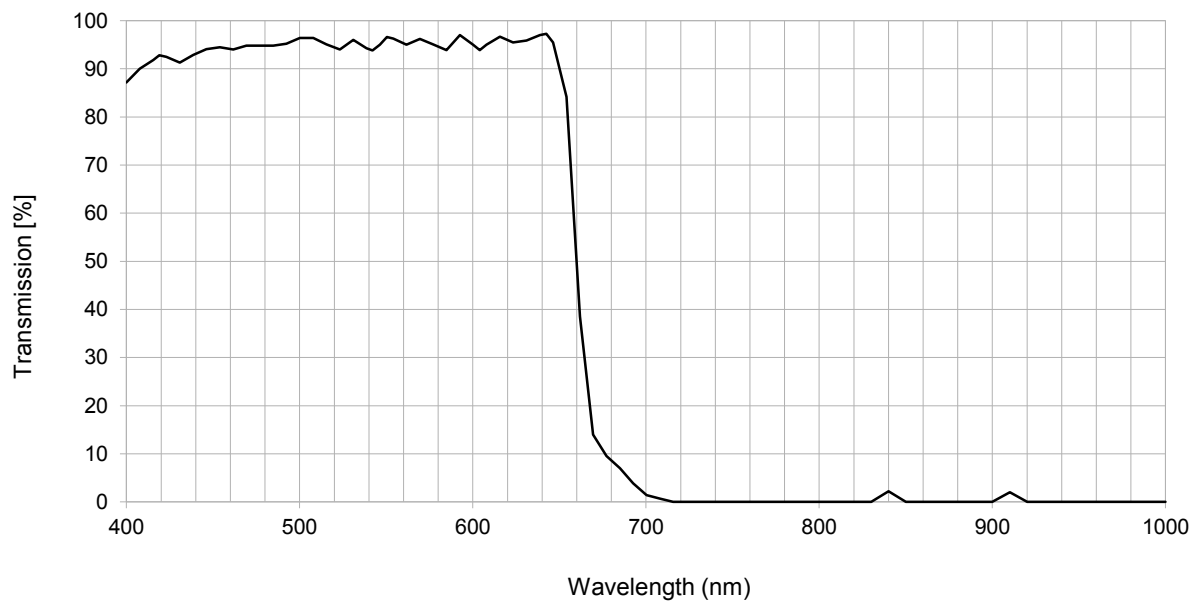


Figure 67: IR-cut filter specification

### 2.3.5 Ingress Protection Class

The camera housing fulfills the *Ingress Protection Class* of IP40. It is protected against solid objects with a diameter larger than 1 mm (tools, wires, and small wires) and has no protection against liquids.

## 2.4 Declarations of Conformity

### 2.4.1 CE

<b>Manufacturer:</b>	Smartek d.o.o Dobrise Cesarica 5 HR-40000 Cakovec Croatia
<b>Product:</b>	Digital Gigabit Ethernet Camera
<b>Type Family:</b>	Giganetix Standard, Giganetix S90 Version, Giganetix Board Level, Giganetix Plus
<b>Type of Equipment:</b>	GC1281M, GC2041C, GC2591M, GC2591C, GC3851M, GC3851C, GC1932M, GC1932C, GC651M, GC651C, GC652M, GC652C, GC653M, GC653C, GC781M, GC781C, GC1031M, GC1031C, GC1391M, GC1391C, GC1392M, GC1392C, GC1621M, GC1621C, GC2441M, GC2441C, GC1021M, GC1021C, GC1291M, GC1291C, GC1601M, GC1601C, GC1921M, GC1921C, GCP1931M, GCP1931C, GCP2061M, GCP2061C, GCP2461M, GCP2641C, GCP1941M, GCP1941C, GCP2751M, GCP2751C, GCP3381M, GCP3381C, GCP4241M, GCP4241C

This equipment is in compliance with the essential requirements and other relevant provisions of the following EC directives:

Reference No.	Title:
89/336/EEC, 92/31/EEC	Electromagnetic Compatibility (EMC directive)

Following standards or normative documents:

EN 55022:1994 Class A + A1:1995 + A2:1997,  
EN 61326:1997 Class A + A1:1998 + A2:2001 + A3:2003,  
EN 55024:1998 + A1:2001 + A2:2003

The equipment specified above was tested conforming to the applicable Rules under the most accurate measurement standards possible, and that all the necessary steps have been taken and are in force to assure that production units of the same product will continue comply with the requirements.



Damir Dolar

Dipl. Ing. Hardware Engineer

Smartek d.o.o.

**2.4.2 FCC**

<b>Manufacturer:</b>	Smartek d.o.o Dobrise Cesarica 5 HR-40000 Cakovec Croatia
<b>Product:</b>	Digital Gigabit Ethernet Camera
<b>Type Family:</b>	Giganetix Standard, Giganetix S90 Version, Giganetix Plus
<b>Type of Equipment:</b>	GC1281M, GC2041C, GC2591M, GC2591C, GC3851M, GC3851C, GC1932M, GC1932C, GC651M, GC651C, GC652M, GC652C, GC653M, GC653C, GC781M, GC781C, GC1031M, GC1031C, GC1391M, GC1391C, GC1392M, GC1392C, GC1621M, GC1621C, GC2441M, GC2441C, GC1021M, GC1021C, GC1291M, GC1291C, GC1601M, GC1601C, GC1921M, GC1921C, GCP1931M, GCP1931C, GCP2061M, GCP2061C, GCP2461M, GCP2641C, GCP1941M, GCP1941C, GCP2751M, GCP2751C, GCP3381M, GCP3381C, GCP4241M, GCP4241C
<b>Directive:</b>	FCC Part 15, Class A

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

Modifications not expressly approved by the manufacturer could void the user's authority to operate the equipment under FCC rules



Damir Dolar

Dipl. Ing. Hardware Engineer

Smartek d.o.o.

### 2.4.3 RoHS II

<b>Manufacturer:</b>	Smartek d.o.o Dobrise Cesarica 5 HR-40000 Cakovec Croatia
<b>Product:</b>	Digital Gigabit Ethernet Camera
<b>Type Family:</b>	Giganetix Standard, Giganetix S90 Version, Giganetix Board Level, Giganetix Plus
<b>Type of Equipment:</b>	GC1281M, GC2041C, GC2591M, GC2591C, GC3851M, GC3851C, GC1932M, GC1932C, GC651M, GC651C, GC652M, GC652C, GC653M, GC653C, GC781M, GC781C, GC1031M, GC1031C, GC1391M, GC1391C, GC1392M, GC1392C, GC1621M, GC1621C, GC2441M, GC2441C, GC1021M, GC1021C, GC1291M, GC1291C, GC1601M, GC1601C, GC1921M, GC1921C, GCP1931M, GCP1931C, GCP2061M, GCP2061C, GCP2461M, GCP2641C, GCP1941M, GCP1941C, GCP2751M, GCP2751C, GCP3381M, GCP3381C, GCP4241M, GCP4241C

This equipment is in compliance with the essential requirements and other relevant provisions of the following RoHS II Directive 2011/65/EU.



---

Damir Dolar



Dipl. Ing. Hardware Engineer

Smartek d.o.o.

## 2.5 List of Supported Features

Series	Giganetix			Giganetix Plus
Interface				
On-Camera Features	GC	GC-S90	GC-BL	GCP
Continuous Streaming (free run)	●	●	●	●
Triggered Operation (single / multi frame)	●	●	●	●
Exposure Control	●	●	●	●
Auto Exposure Control	● <sup>1</sup>	● <sup>1</sup>	● <sup>1</sup>	● <sup>2</sup>
Frame Rate Control	●	●	●	●
Partial Scan (ROI / AOI / WOI)	● <sup>3</sup>	● <sup>3</sup>	● <sup>3</sup>	●
Multiple ROI	○	○	○	● <sup>4</sup>
ROI Centering	● <sup>5</sup>	● <sup>5</sup>	● <sup>5</sup>	●
Binning	● <sup>6,10</sup>	● <sup>6,10</sup>	● <sup>6,10</sup>	● <sup>7</sup>
Reverse X (Horizontal Mirroring)	● <sup>5</sup>	●	● <sup>5</sup>	● <sup>4</sup>
Reverse Y (Vertical Mirroring)	● <sup>5</sup>	●	● <sup>5</sup>	● <sup>4</sup>
Analog Gain Control <sup>8</sup>	●	●	●	●
Auto Analog Gain Control	● <sup>1</sup>	● <sup>1</sup>	● <sup>1</sup>	● <sup>2</sup>
Analog Black Level Control <sup>8</sup>	●	●	●	●
Online Tap Balancing <sup>9</sup>	●	●	●	○
Factory Tap Calibration <sup>9</sup>	○	○	○	●
Gamma	● <sup>5</sup>	● <sup>5</sup>	● <sup>5</sup>	●
Digital Shift	● <sup>5</sup>	● <sup>5</sup>	● <sup>5</sup>	●
Lookup Table	● <sup>1</sup>	● <sup>1</sup>	● <sup>1</sup>	● <sup>2</sup>
Chunk Data	○	○	○	●
Software Trigger	●	●	●	●
External Trigger (Line)	●	●	●	●
Line Debouncer (Trigger)	●	●	●	●
Line Input Delay (Trigger)	●	●	●	●
Configuration Storing (User Sets)	●	●	●	●
Acquisition / Exposure / Frame Active (Output)	●	●	●	●
Acquisition / Frame Trigger Wait (Output)	●	●	●	●
User Defined Outputs	●	●	●	●
IP Configuration (LLA / DHCP / Persistent)	●	●	●	●
Jumbo Frame Size (in Bytes)	4000	4000	4000	8000
Inter Packet Delay	●	●	●	●

Table 42: Camera / API feature list (1/2)

Series	Giganetix			Giganetix Plus
Interface				
On-Camera Features	GC	GC-S90	GC-BL	GCP
Frame Transfer Delay	●	●	●	●
Time Stamps	●	●	●	●
Pixel Data Formats				
Mono8	●	●	●	●
Mono10Packed	● <sup>1</sup>	● <sup>1</sup>	● <sup>1</sup>	● <sup>11</sup>
Mono12Packed	○	○	○	● <sup>11</sup>
Mono16	●	●	●	●
Bayer8	●	●	●	●
Bayer16	●	●	●	●
Hardware				
Housing	●	●	○	●
Angled Sensor Head (90°)	○	●	● <sup>12</sup>	○
Inputs (opto coupled)	2	2	2	2
Outputs (opto coupled)	2	2	2	2
Power over Ethernet	●	○	●	●

<sup>1</sup> GC1932 mono models supported only

<sup>2</sup> GCP mono series only

<sup>3</sup> On multi-tap CCD sensors supported only in vertical direction

<sup>4</sup> GCP1931 models supported only

<sup>5</sup> GC1932 models supported only

<sup>6</sup> Not supported by models with color CCD, multi-tap CCD and individual CMOS sensors

<sup>7</sup> GCP1931 models supported only (Horizontal only)

<sup>8</sup> Overall and tap independent, separate color channels only on CMOS models

<sup>9</sup> Multi-tap CCD sensors only

<sup>10</sup> GC1932 models supported only (Horizontal only)

<sup>11</sup> Available only on Mono GCP cameras

<sup>12</sup> Flexible positioning, sensor head connected via FPC cable to mainboard only

Table 43: Camera / API feature list (2/2)

### 3 Smartek GigEVisionSDK Library

The SMARTEK Vision GigEVisionSDK provides a set of tools, guides and samples, useful for the configuration and image acquisition from GigE Vision cameras, as well as the integration into own software applications. The GigEVisionSDK library consists of 4 basic parts:

- **Smartek Filter Driver** - SMARTEK Vision provides its own filter driver to ensure optimal performance of the digital camera. This driver is compliant to the GigE Vision standard. It separates incoming packets containing image data from other traffic on the network, optimizing the image data flow from the camera to the software.
- **GigEVisionClient** - The GigEVisionClient is an as binary and source available sample application, which contains and utilizes the whole function set available by the GigEVisionSDK API in an intuitive graphical user interface. Besides displaying images grabbed from the cameras, it provides graphical access to e.g. multiple cameras, their configuration and available post processing functions.
- **GigEVisionSDK API** - The GigEVisionSDK offers an application programming interface (API) for for Gigabit Ethernet Vision (GigE Vision) cameras. It supports the programming languages C/C++, Delphi, C# and VB .NET, and allows an easy integration of a SMARTEK Vision camera in own software applications.
- **ImageProc API** - The ImageProc API extends the basic camera functionality, provided by the GigEVisionSDK API, by color and post processing functions like e.g. debayering, gamma, look-up table (LUT) and color correction algorithms. All programming languages supported by the GigEVisionSDK API are supported by the ImageProc API as well.

#### 3.1 Supported Operating Systems

The SMARTEK Vision GigEVisionSDK has been created to support Microsoft Windows as well as Linux operating systems. For Microsoft Windows, one software installer supports all versions and system architectures (32 or 64 Bit). The GigEVisionSDK for Linux is available in versions for Debian and RPM based packet managers, separately for 32 and 64 Bit. Table 44 contains a list of the supported operating systems and the appropriate installation package.

	Operating System	32 Bit	64 Bit	SDK Package
Microsoft	Windows XP	✓	✓	Windows Executable
	Windows Vista	✓	✓	
	Windows 7	✓	✓	
	Windows 8	✓	✓	
	Windows 8.1	✓	✓	
Linux	DEB based (Debian, Ubuntu, Knoppix)	✓	✓	32Bit.deb / 64Bit.deb
	RPM based (Fedora, Red Hat, SUSE)	✓	✓	32Bit.rpm / 64Bit.rpm

Table 44: Supported operating systems and installation packages

### 3.2 (Un-)Installing the GigEVisionSDK on Microsoft Windows and Linux

QuickStart and Installation guides for Microsoft Windows and Linux operating systems can be downloaded separately from the SMARTEK Vision webpage:

[www.SMARTEKvision.com/downloads.php](http://www.SMARTEKvision.com/downloads.php)

The uninstallation on Microsoft Windows can be started by removing the software in *Programs and Features* in the Microsoft Windows *Control Panel*.

### 3.3 Unattended Installation on Microsoft Windows Operating Systems

In cases where the GigEVisionSDK needs to be included into user's own installer it is possible to install the package in silent mode. This way it is possible to install the package without any graphical output and user interactions.

To run the installation silently, execute the binary from the command prompt as Administrator, with the following flags (in one line):

```
SMARTEK_Vision_GigEVisionSDK_Vxxxx.exe /VERYSILENT  
/SUPPRESSMSGBOXES  
/NORESTART  
/TYPE="minimal"
```

There are three types of installations that can be performed:

1. /TYPE="minimal" - minimal installation (binaries only)
2. /TYPE="compact" - compact installation (binaries, headers, docs and samples)
3. /TYPE="full" - full installation (binaries, headers, docs, samples and sources)



### 3.4 Manual Filter Driver Installation / Uninstallation

If the SMARTEK Vision GigEVision Filter Driver needs to be installed manually, it can be installed independently of the GigEVisionSDK by executing a batch script (\*.bat), located in the SDK's installation directory after successfully installing the GigEVisionSDK.

This is for example the case if the first instance of the GigEVisionClient starts without the filter driver, showing the message in Figure 68. Additionally a warning message in the top bar of GigEVisionClient is displayed - "Warning: Smartek Filter Driver not loaded".

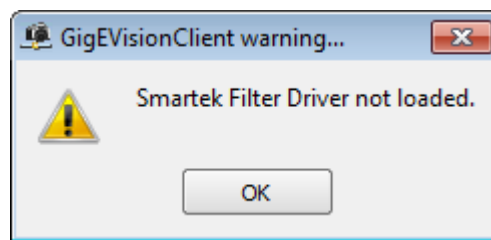


Figure 68: Warning: Smartek Filter Driver not loaded

The batch scripts can be found in a subfolder of the installation directory. By default the GigEVisionSDK is installed into the following folder:

```
C:\Program Files\SMARTEKvision\GigEVisionSDK\..
```

The driver specific files are located in the following subfolder:

```
..\drivers\FilterDriver\..
```

The driver is being installed by executing **GigEVDrvInstall.bat**, uninstalled by executing **GigEVDrvUninstall.bat**

### **3.5 User Buffer**

By default, memory for camera raw images is allocated by driver in kernel space. User application can provide own allocated memory to be used instead. User buffer example can be found in the API documentation located in the GigEVisionSDK installation folder.

### 3.6 GigEVisionClient

The GigEVisionClient is a Qt-based open-source application installed along with the GigEVisionSDK. It utilizes and demonstrates the major function set available by the API in an intuitive graphical user interface and is capable of acquiring and controlling single or multiple GigE Vision compliant cameras.

After the installation of the GigEVisionSDK the GigEVisionClient can be started by the appropriate shortcut in the Microsoft Windows Start menu (All Programs ⇒ SMARTEK Vision). The binaries can be found within the installation directory, usually located at:

```
C:\Program Files\SMARTEKvision\GigEVisionSDK\bin\
```

The source code is located at:

```
C:\Program Files\SMARTEKvision\GigEVisionSDK\src\GigEVisionClient
```

#### 3.6.1 Graphical User Interface (GUI)

Figure 69 shows the default GUI elements of the GigEVisionClient.

- **Menu bar** - always on top of the application. It provides access to all functionalities available on the toolbar. Also main dialogs within the GigEVisionClient can be switched on or off under the entry Control. Several dialogs are disabled by default and can be activated manually:
  - *Preview* - separate displays for each connected camera
  - *Device Property Info* - GenICam attributes of the selected device property
  - *API Settings* - access to configuration settings of API and driver
  - *Histogram* - display a histogram for the currently selected device
  - *Log* - display the API log
- **Toolbar** - enables quick access to basic functions of the camera (find, connect, disconnect, IP setup), image handling (open, save, zoom etc.), GUI handling (save GUI arrangement, open, reset GUI to default etc.).
- **Device list dialog** - lists all GigE Vision compliant devices found on the network and its connection status. It further acts as the camera selector.
- **Device properties dialog** - gives access to all features (GenICam) supported by the device.
- **Image Processing properties dialog** - gives access to the parameterizations settings of the image processing algorithms.
- **Info bar** - displays information like image size, frame rate, data transfer rate, cursor position and pixel value at cursor position.
- **Image display window** - main window for displaying a single image or video stream.

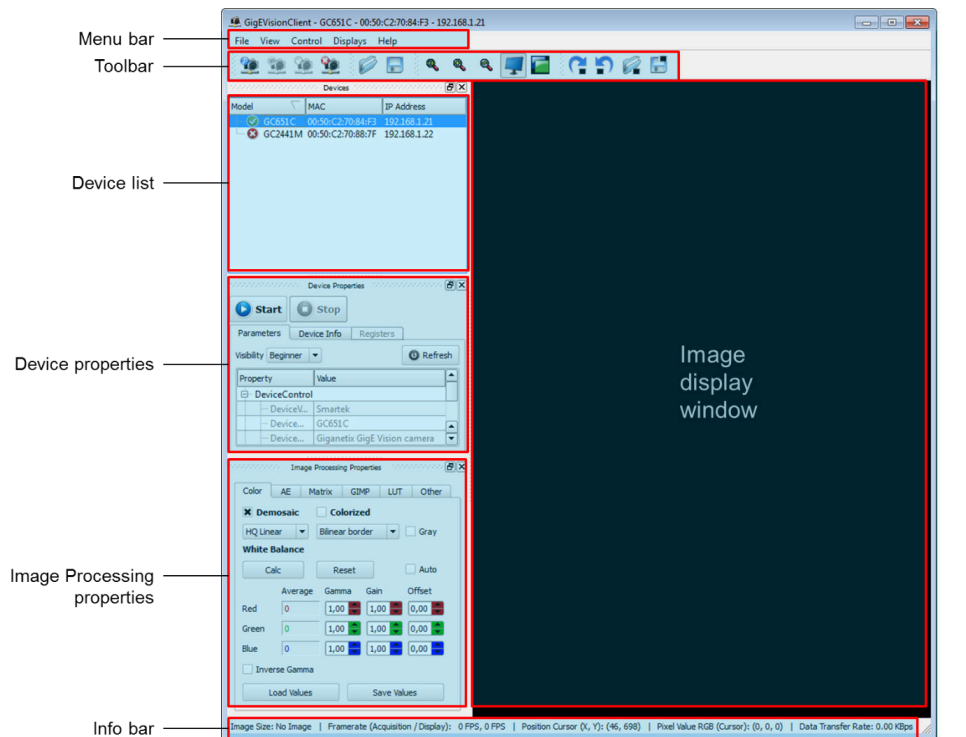


Figure 69: GigEVisionClient Graphical User Interface (GUI)

Refer to Figure 70 for a full description of all GUI elements on the toolbar.

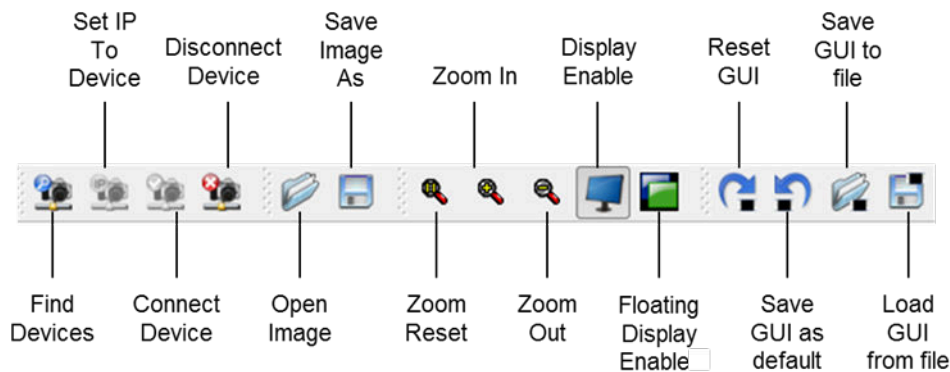


Figure 70: Toolbar description

Due to the possibility that all the dialogs within the GigEVisionClient are dockable, the user can set his own user interface as default or save it to a file, so that his own GUI arrangement can be loaded to the GigEVisionClient installed on other machines. The GUI save and reset features are accessed through the menu bar or the toolbar, as shown in Figure 70.



**Note**

The Floating Display feature allows the user to arrange different image displaying windows for each camera's video stream on the screen.

### 3.6.2 Acquire Images from Camera(s)

In this section a step-by-step guide will be introduced, showing how the user can start the image acquisition from a camera using the SMARTEK Vision GigEVisionClient.

#### 3.6.2.1 Device Enumeration

After the GigEVisionClient is started, it automatically searches for GigE Vision compliant devices connected to the network. All found devices are listed in the device list dialog. If the required camera is attached or powered-up subsequent and is not shown in the list, the user can manually update the list and search for devices.

To search for devices within the network(s) the computer is connected to, click the *Find Devices* icon in the *Toolbar*, shown in Figure 71.

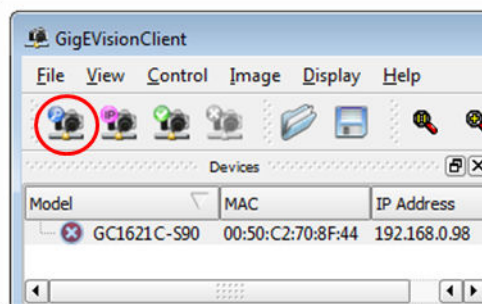


Figure 71: Find Devices icon



#### Note

If none of the connected cameras was found, check the status of the network adapters and their configuration, as well as the status LEDs on the Ethernet connector of the camera, described in chapter 2.3.1 - *Ethernet Interface*. Make sure everything is plugged properly and that the firewall settings are not blocking the connection to camera or GigEVisionClient.

After discovering one or multiple cameras, there is one of three available flags in front of each camera name displayed in the *Devices list*:




Device available and waiting for connection



Connection to device established



Warning

In case the  *Warning* sign is shown in front of a camera name, there could be two reasons:

1. The IP-address and subnet mask of network interface card or camera are invalid
2. The link speed of the used network interface is providing less than 1000 Mbit/s

For a quick start it is recommended to only use Gigabit Ethernet NICs, configured according to Table 73 shown in chapter 5.2 - *LAN IP Configuration*.

In all cases it must be ensured that further NIC's within the PC are not configured for an IP-address within the same logical network as the NIC for the camera connection.

### 3.6.2.2 Device IP Setup

To change the IP address of a camera, select the target device in the list of devices and press the *Set Ip to Device* icon shown in Figure 72.

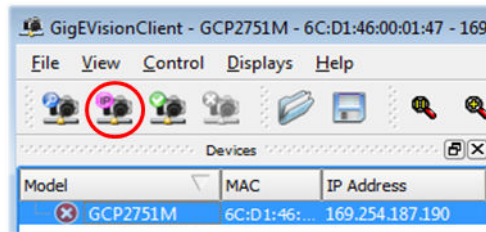


Figure 72: Set Ip To Device icon

A new window will open showing an access mask to the IP address, subnet mask and gateway configuration of the chosen camera. Make sure that the target IP address is not applied to another device in the network.

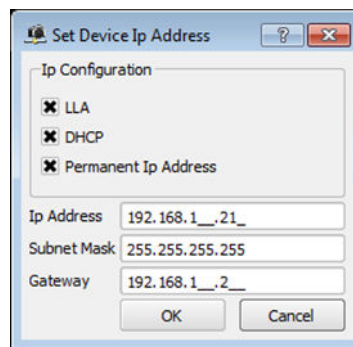


Figure 73: Set Ip To Device dialog

After fixing a valid IP address configuration to the camera, the warning symbol next to the camera model name will change like shown in Figure 74, the *Connect Device* icon can now be used to connect to the selected camera.

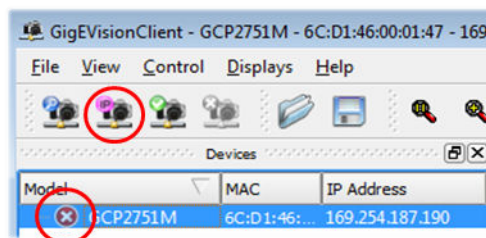


Figure 74: Connect Device icon

### 3.6.2.3 Device Properties

The *Device Properties* dialog contains all information and settings of the chosen camera, provided by the camera's GenICam file.

General informations about the camera selected from the list of discovered devices are displayed in the tab *Device Info*. Among others it contains the device name, firmware and hardware versions as well as the current IP configuration. It is accessible already before a connection has been established.

The *Parameters* tab shown in Figure 75 shows the parameters of the camera and is only accessible while a connection to the camera is established. It displays a tree of features extracted from the GenICam description file of the camera and enables the adjustment of camera settings according to the needs of the application.

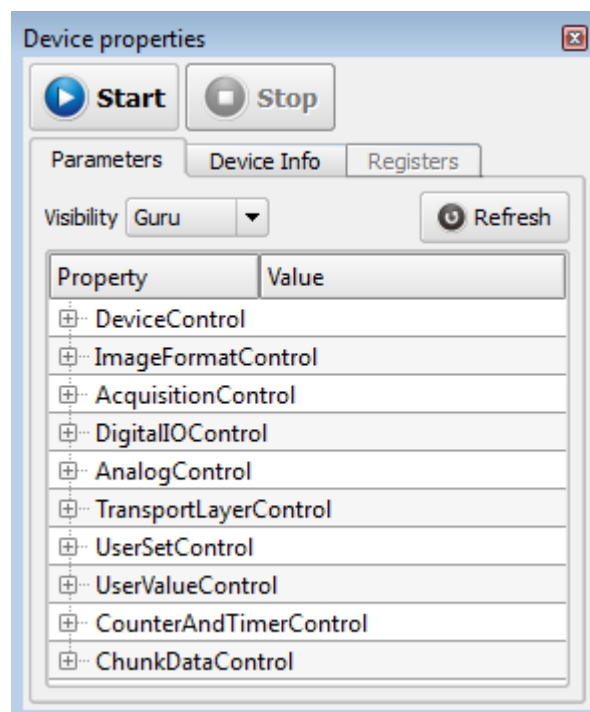


Figure 75: Device Properties - Parameters tab

According to the current settings of the camera, the acquisition can be started by pressing the *Start* button shown in Figure 76. To receive a continuous image stream from the camera, without having any external trigger signals applied, it must be ensured that the *AcquisitionMode* is set to *Continuous* and the *TriggerMode* is set to *Off*.

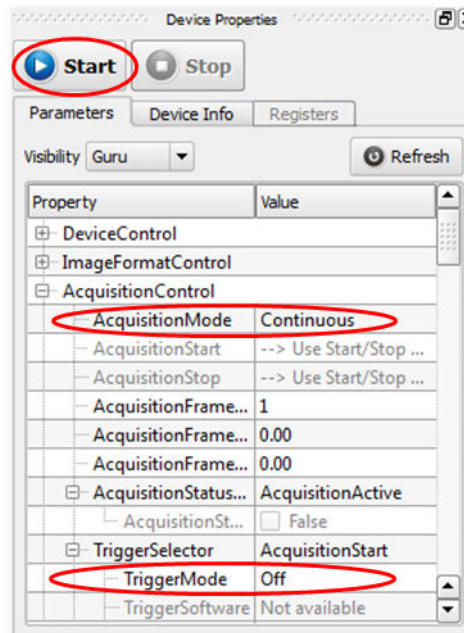


Figure 76: Device Properties - Starting a continuous stream

A running acquisition can be quit by pressing the *Stop*-button. Multiple acquisitions can be started in parallel by choosing further cameras, the output of the currently selected device is shown in the *Image Display* window.



### 3.6.2.4 Multiple Devices, Multiple Views

The GigEVisionClient supports the operation of multiple devices that are detected and connected to the network in parallel. The video stream of each device can be displayed in separated floating windows or grouped together in a *Preview* dialog. Figure 77 demonstrates these possibilities with two connected cameras. On the right the "Preview" dialog contains the video streams of the two cameras. This dialog can be enabled through *Control* ⇒ *Preview* in the menu bar. The floating displays are accessible through the menu *Displays* in the menu bar.

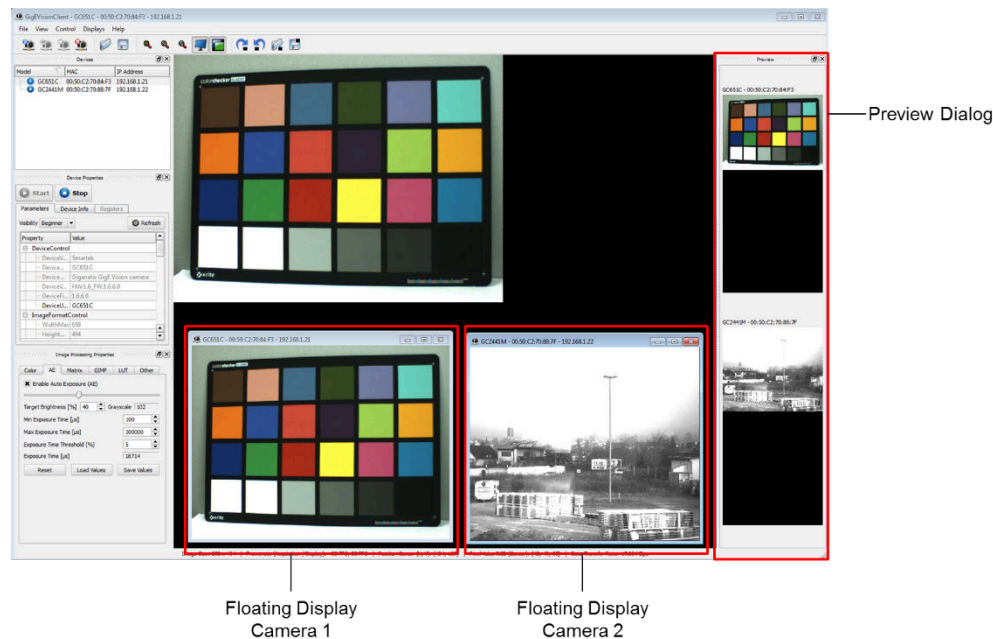


Figure 77: Preview Dialog and Floating Displays for multiple cameras



#### Note

The maximum number of devices depends on the memory and performance of the host PC, as each of the devices will cause an amount of load to the system.

### 3.6.2.5 Image Processing

Image processing algorithms provided by the ImageProc API can be accessed within the *Image Processing* dialog, shown in Figure 78. It enables the user to apply and parameterize the available image preprocessing functions.

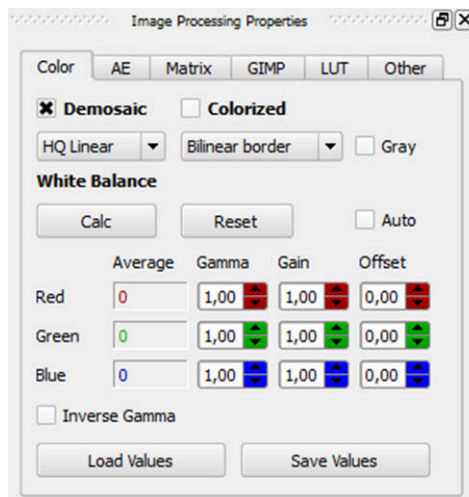


Figure 78: Image processing properties - Color tab

Table 45 shows an overview of the tabs and functions each tab includes. A deep description of all listed image functions can be found in chapter 7 - Image Processing in *GigEVisionSDK*.

Tab	Functions	Comment
Color	Demosaicing, white balancing, gamma, gain and offset correction	
AE	Auto Exposure	
Matrix	3x3 Matrix multiplication including value storing/loading	Only available for color cameras
GIMP	GIMP based color manipulation via hue, lightness and saturation	Only available for color cameras
LUT	Look Up Table generation, application and storing/loading	
Other	Image sharpening, Image Flip/Rotate	

Table 45: *GigEVisionClient* - Image processing functions

### 3.6.3 API Settings Dialog

The *API Settings* dialog, accessible after activating it in the *Control* menu of the *Menu Bar*, displays various settings of the API, driver and memory management. The *API* tab, shown in Figure 79, allows the modification of all available API parameters and gives access to the packet resend settings and statistics of the current driver session, as well as it enables the user to rise or decrease the image buffer within the camera driver. This buffer represents space in the non-paged memory pool of Windows and can be used to improve the performance when loosing images because of low system performance.



#### Note

Since the amount of memory in the non-paged memory pool is limited (Windows XP limits it to 256MB, Windows 7 limits it at 75% of physical RAM), the number of images in the image buffer must be calculated correctly.

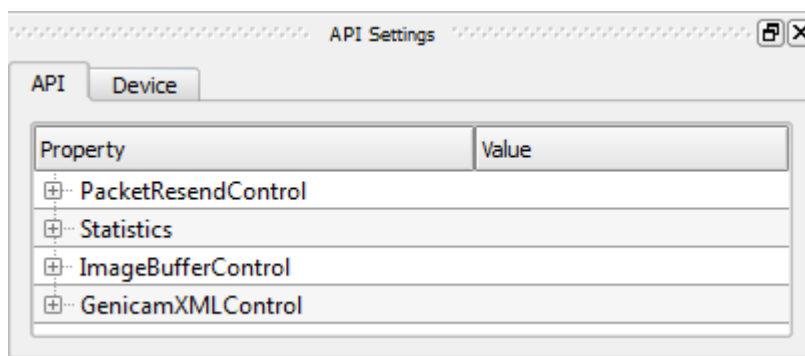


Figure 79: API Settings

Additionally the *Device* tab of the *API Settings* dialog provides access to the packet statistics on device side.

### 3.6.4 Chunk Data Control

To access chunk data control in GigEVisionClient visibility must be set to Expert. Controls are located at bottom of Device Properties window. Figure 80 shows chunk data control.

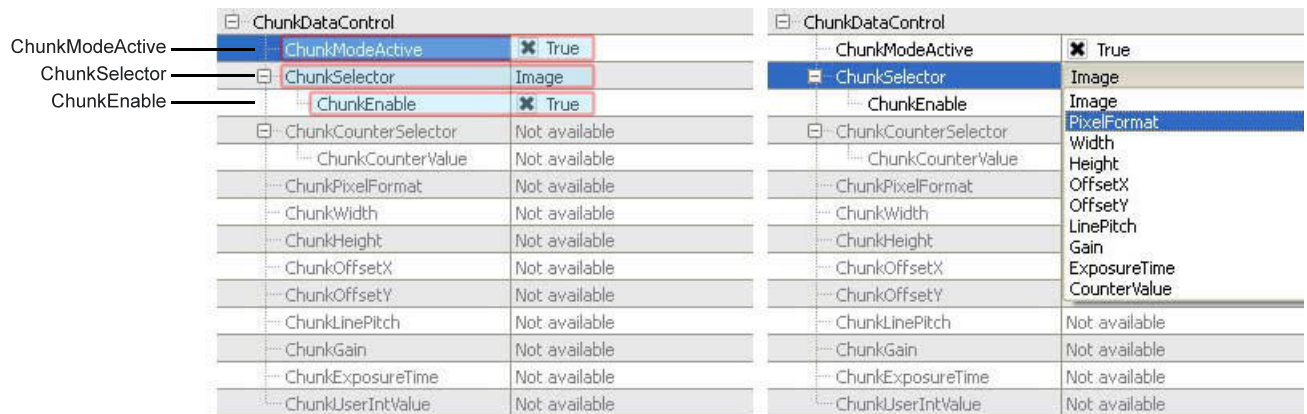


Figure 80: Chunk Data Control

- **ChunkModeActive:** enable or disable chunk data ( if chunk data is disabled then normal image is sent )
- **ChunkSelector:** select chunk that will become active
- **ChunkEnable:** enable or disable active chunk ( mandatory chunks cannot be disabled )

When chunk mode is enabled and acquisition is started, enabled chunks will be displayed in new tab that appeared under Image Processing Properties. Figure 81 shows chunk data values tab.

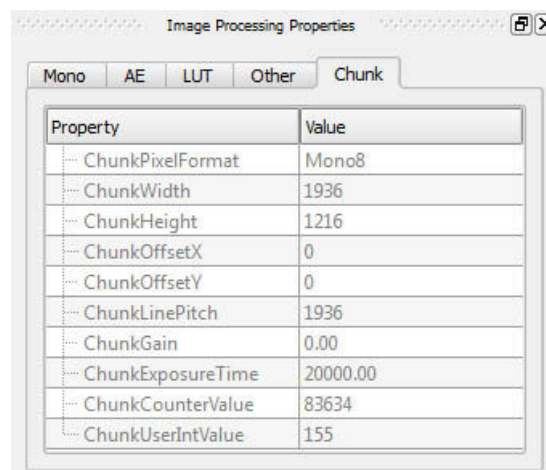


Figure 81: Chunk Data Values

### 3.6.4.1 Setting Chunk Data values

UserIntValue can be set up through properties window shown in Figure 82. The value is included in Data Payload if UserIntValue chunk is enabled.

UserValueControl	
UserIntValue	0

Figure 82: Chunk Data Values

Figure 83 shows CounterAndTimerControl.

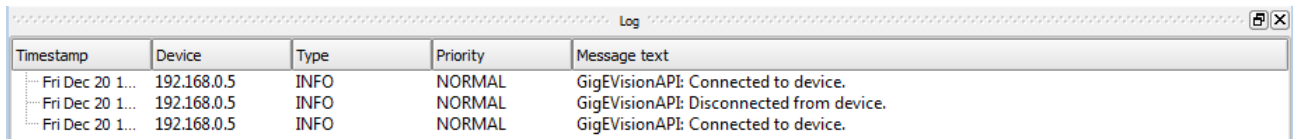
CounterAndTimerControl	CounterAndTimerControl	
CounterSelector	CounterSelector	Counter1
CounterEventSource	CounterEventSource	FrameStart
CounterValue	CounterValue	0

Figure 83: Chunk Data Values

- **CounterAndTimerControl:** Group for counter and timer controls
- **CounterSelector:** Select which counter will be active
- **CounterEventSource:** Select source for counter
- **CounterValue:** Set / read counter value

### 3.6.5 Log Dialog

The Log dialog contains logging information from SDK and cameras, shown in Figure 84.



Timestamp	Device	Type	Priority	Message text
Fri Dec 20 1...	192.168.0.5	INFO	NORMAL	GigEVisionAPI: Connected to device.
Fri Dec 20 1...	192.168.0.5	INFO	NORMAL	GigEVisionAPI: Disconnected from device.
Fri Dec 20 1...	192.168.0.5	INFO	NORMAL	GigEVisionAPI: Connected to device.

Figure 84: Log dialog with API logging

### 3.6.6 Firmware Update

The *GigEVisionClient* contains a module to update the firmware of a Giganetix camera. To update the firmware of a SMARTEK Vision camera, choose and connect the target camera in the GigEVisionClient and start the *Firmware Update* dialog via the *Control* category in the menu bar.

In the Firmware Update Dialog, shown in Figure 85, follow the following steps:

- Browse for Firmware Updates**

Find and open a firmware to be uploaded to the camera by pressing the *Browse* button. The latest firmware can be requested from your local sales representative or at [support@SMARTeKvision.com](mailto:support@SMARTeKvision.com).

- Compatibility Check Passed**

After selecting and opening a firmware file, the application will run a compatibility test between the device and firmware. If the selected firmware is compatible to the selected camera, the shown text is tagged as *"PASSED"* and the *Upload new firmware to device* button will become available.

- Flash Firmware**

To upload and flash the firmware to the target camera, press the *Upload new firmware to device* button.

- Successful Update**

In the last step of the firmware update the camera is restarted by the application. If this step was executed successfully, the update window can be closed and the camera is ready to be operated with the new firmware.

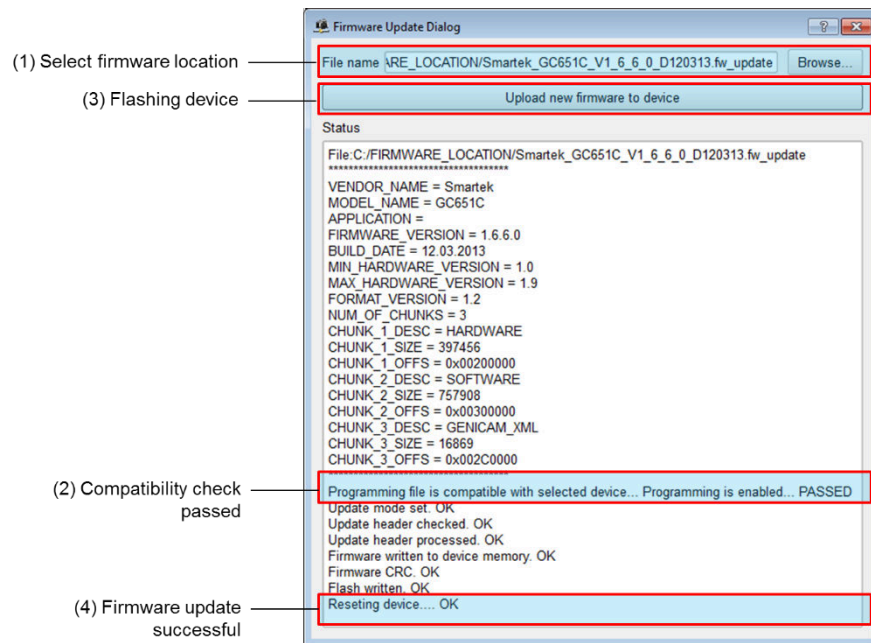


Figure 85: Firmware update dialog after the update is successfully executed



**Note**

In case of any major errors during the update process, please repeat the firmware upload. Do not restart a camera before the process was finished successfully!

## 4 Image Acquisition

The following chapter gives a brief overview about the general principles of digital image acquisition based on the Giganetix series, starting at the point where the image was projected to the image sensor plane. It further includes a description of the camera's main functionality as well as the main configuration options of the image acquisition.

### 4.1 General Camera Architecture

The Giganetix camera series consist of multiple electronic components to convert incoming light to a digital signal, process it and send it to the host device or network. Figure 86 shows the simplified architecture of each camera, containing the image data path (orange) from the sensor to the Ethernet Network, as well as multiple control paths (red).

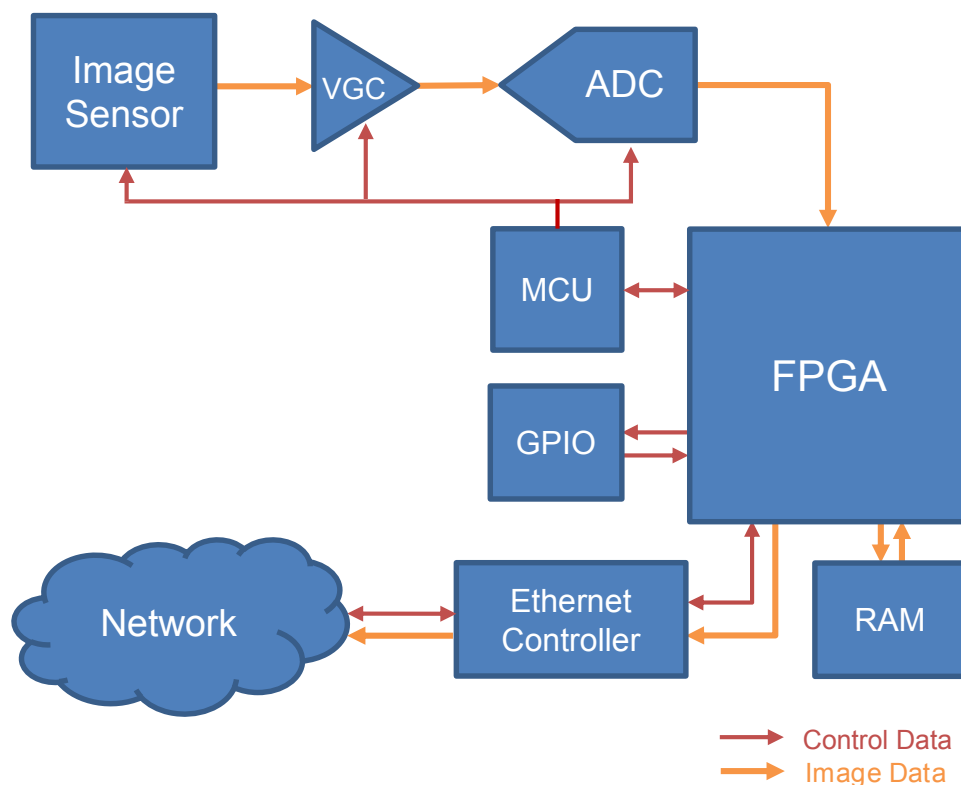


Figure 86: Camera Architecture Scheme

Like shown in Figure 86, the image data chain (orange) starts with the *Image Sensor* converting incoming light to electrical load. On usual area image sensors, over a million pixels are arranged in a two dimensional array on which the light of a scene is projected through a lens. Each pixel captures the photons of the light at a different point of the scene, making it possible to detect the whole scene within a defined grid. Hitting the sensor's active pixel array, photons generate electrons which in turn create an electrical load in each pixel, defining the intensity of each. The amount of electrical load depends on the strength of the light and the time frame for which the pixels are exposed to light, the so called *Exposure* or *Integration Time*.



Before the load of each pixel can be digitized, it needs to be amplified by a *Variable Gain Control* (VGC) circuit according to the input characteristics of the *Analog-to-Digital Converter* (ADC). The amplification factor of this component can be controlled via the *Analog Gain* through the camera control interface.

By default the signal is amplified as strong as just needed to completely drive the *ADC* when the pixels reach their full well capacity. Higher gain values will result in a cut off signal at high signal levels, lower gain values will not amplify the signal enough to ever reach a saturated digital image, even if the sensor reached its saturation. To improve the signal quality and remove a possible dark noise from the image, which is e.g. created by thermal influence, a so called *BlackLevel* value is globally subtracted from each pixel as well.

After the preparation, the signal of each pixel is digitized by the analog-to-digital converter. The bit depth of each digital pixel value is defined by the resolution of the ADC used, typically in range from 8 to 16 bits. All further image processing is done based on the digital signals and takes place in the camera's *FPGA*. The external memory (*RAM*) connected to the *FPGA* is used to buffer the image data, decoupling the data processing domain from the data transmission domain (Ethernet).

In the last step of the image processing chain, the final image data is passed to the *Ethernet Controller*. Here the data is segmented into GigE Vision compliant Ethernet packets which are sent over the Ethernet physical interface to the network or capture devices.

### 4.1.1 CCD Sensor Readout

The Giganetix camera family is equipped with a selection of CCD sensors from Sony and Truesense Imaging. Figure 87 brochures the Interline Transfer technology used in all of the equipped CCD sensors, as well as the individual camera front-end.

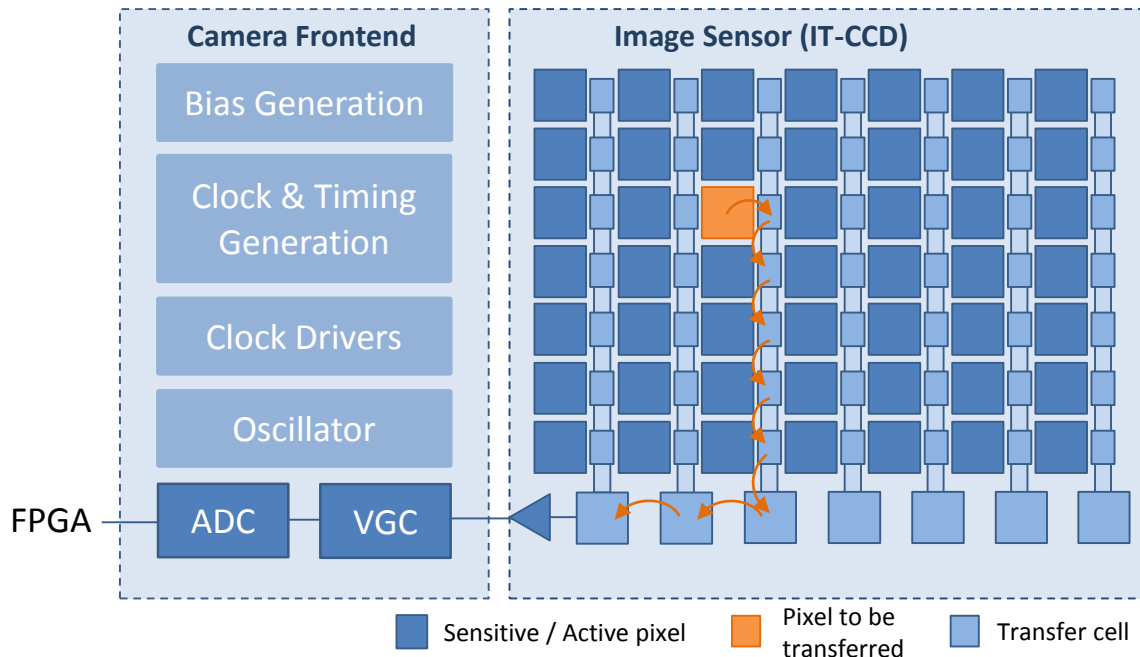


Figure 87: Giganetix Frontend with Single Tap Interline Transfer CCD

After the period of exposure, all collected charges are transferred at the same time from the active pixels into vertical shift registers, located beside each column of active- or photosensitive cells. As the transfer cells are not photosensitive, no mechanical shutter is needed. As all pixels are read out at the same time, Interline Transfer CCD sensors have a Global Shutter as well.

The charges from top of the vertical shift registers are moved line by line down to the horizontal shift register, shown in the bottom of the active array. From the horizontal shift register, charges are converted to voltages and moved out of the image sensor (from right to left). On the camera frontend, containing the sensor, the image signal is amplified (VGC) and digitized (ADC) and provided to the FPGA for further processing.

#### 4.1.2 Multi-Tap CCD Sensor Readout

In contrary to classic Single Tap CCD sensors, the pixel array of Multi Tap CCDs is read out from several sides. The big advantage of this approach of parallel readout is the multiplied amount of pixel data which can be read at the same time. Depending on the count of taps, multi-tap sensors can easily reach a multiple of frame rates compared to single tap sensors.

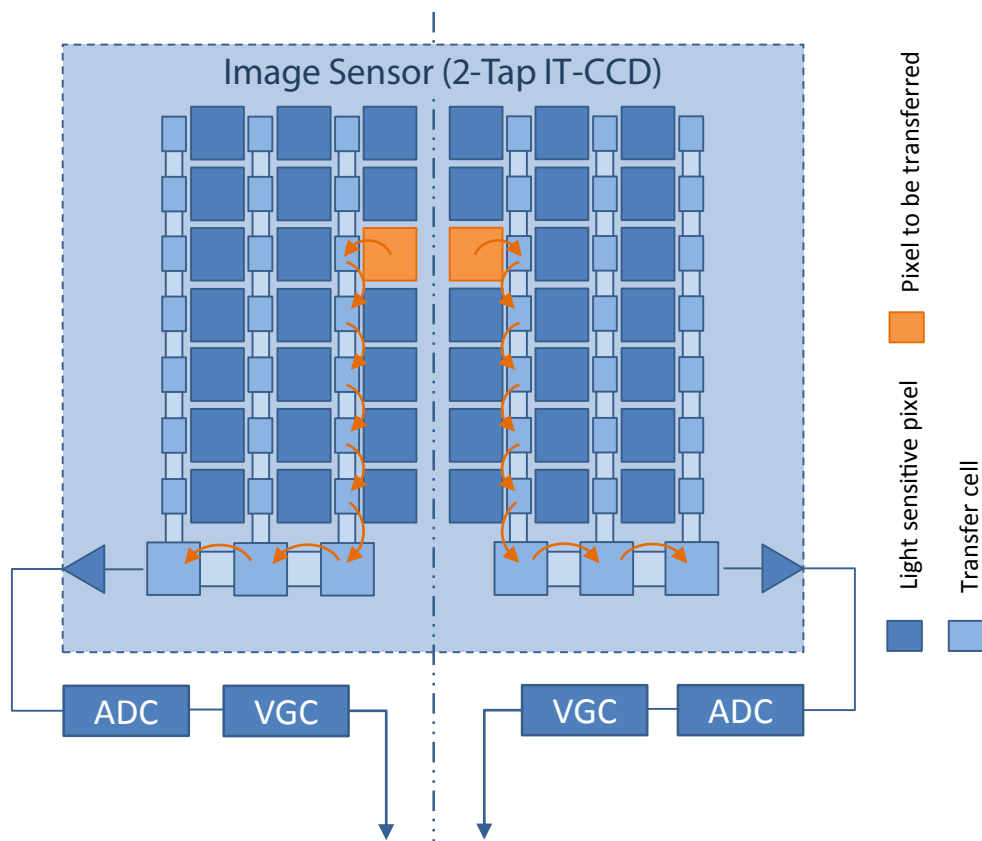


Figure 88: Dual Tap Interline Transfer CCD

Like shown in Figure 88, where the schematic of a dual tap or 2-tap sensor is shown, the active pixels are read out in the same way like on a Single Tap CCD, but from two sides. This mechanism leads to a doubled frame rate compared to an equally clocked single tap CCD, but also to an issue.

As both arrays are read out synchronously, the pixel data of each tap has to be amplified and digitized by an individual circuit. As electrical components are subjected to tolerances, each circuit behaves slightly different over temperature. This causes in slightly dissimilar brightness levels between the taps, shown in Figure 89 (left), which need to be matched for a proper functioning of the camera depending on the actual device temperature (right).



Figure 89: 2-Tap Sensor - Unbalanced image (left) and matched image (right)



#### Note

The cameras of the Giganetix family contain two mechanisms to appropriately match the sensor taps; cameras of the Giganetix series (Standard, GC-S90 and GC-BL) provide by default an automatic online matching which is described with its configuration parameters in chapter 4.3.4 - *Automatic Tap Balancing*. All cameras of the Giganetix Plus series (GCP) are factory calibrated in their specified temperature range, configuration by the user is not necessary.

### 4.1.3 CMOS Sensor Readout

A CMOS sensor reads the accumulated charge of each cell in the image individually, where it was already converted to a voltage. There are several transistors at each pixel which do the conversion and make each pixel be addressable by the horizontal and vertical circuit, using more traditional wires. Because of the high demand for space by additional transistors on each pixel, the light sensitivity of a CMOS chip tends to be lower, as the photosensitive area shrinks with the amount of transistors.

Each pixel is read out and reset separately after each other. On Global Shutter CMOS sensors the charge of each pixel is additionally buffered in a non-photosensitive area of the pixel before, while on Rolling Shutter sensors the charge is read out directly from the exposed pixel. This postpones the moment of readout and thus shifts (Electronic Rolling Shutter) or extends (Global Reset Release) the duration of exposure from pixel to pixel (and line to line). Both architectures have their advantages and disadvantages; while a Rolling Shutter has problems in motion scenes due to the fact that the lower lines of the images are later exposed than the top ones, Global Shutter sensors show up additional noise and lower sensitivity due to their higher amount of transistors per pixel. In case of a Rolling Shutter the mentioned effect can be removed by using strong synchronized strobe illuminations or a mechanical shutter.

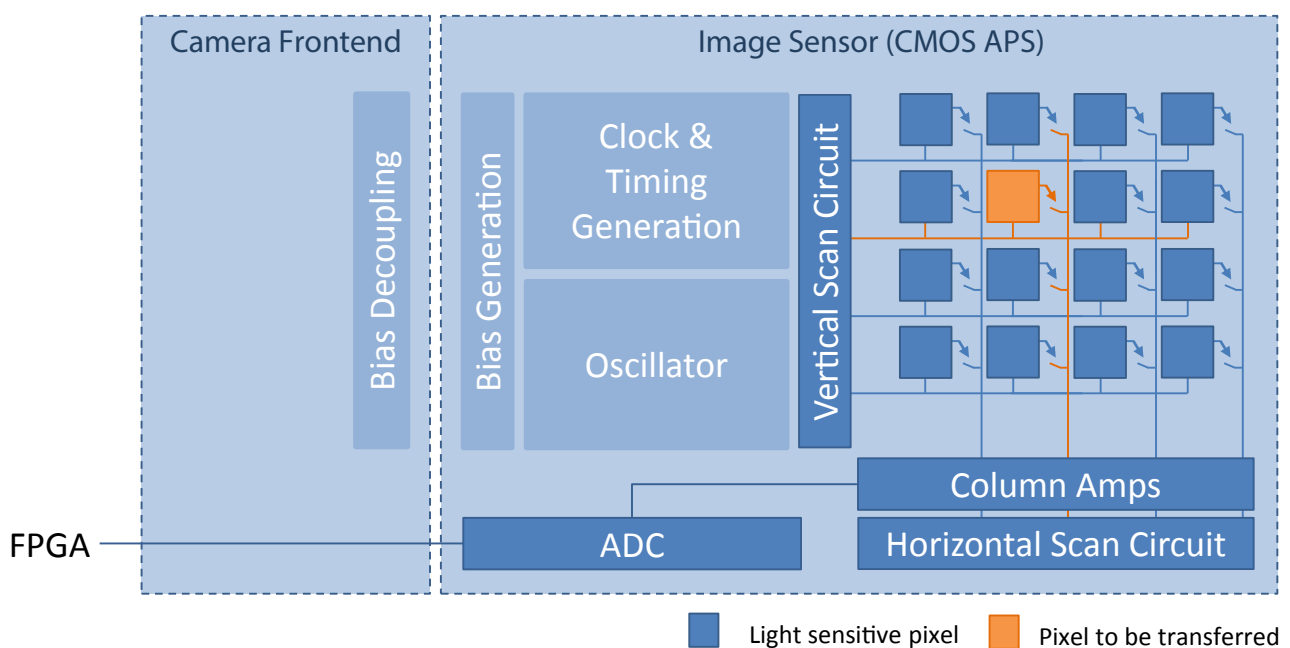


Figure 90: Giganetix Frontend with CMOS Active Pixel Sensor with integrated Gain and ADC

As shown in Figure 90, on CMOS image sensors the image data is already amplified in the sensor's *Column Amps* and digitized by the *ADC* before leaving the image sensor. Depending on the sensor type, also additional processing can already take place within the sensor. The output of the CMOS image sensors used is a digital signal which can directly be forwarded to the camera's FPGA. The electronics in the camera's frontend are mainly dedicated to provide clean and separated supply powers for the sensor and its periphery, and route the sensor control bus (I<sup>2</sup>C).

#### 4.1.4 CCD vs. CMOS - Sensor Performance

Both CCD and CMOS technologies use the same principle, they transform light into electric charge and convert it into electronic signals.

In a CMOS sensor, each pixel has its own charge-to-voltage conversion, and the sensor often also includes amplifiers, noise correction, and digitization circuits, so that chip outputs are digital bits. In a CCD sensor, each pixel's charge is transferred through a very limited number of output nodes to be converted to voltage, buffered, and sent off chip as an analog signal.

This difference in readout techniques has major impact on sensor limitations and capabilities. Eight properties describe sensor performance:

1. **Speed** - an attribute that favors CMOS over CCDs because most of the camera functions can be placed on the image sensor.
2. **Quantum Efficiency** - the ratio between output signal and unit of input light energy. Rolling Shutter CMOS sensors caught up massively within the past years and offer a similar performance.
3. **Uniformity** - is the consistency of response for different pixels under identical illumination conditions. CMOS were traditionally much worse than CCDs, however new amplifiers have made the illuminated uniformity of some CMOS close to that of CCDs.
4. **Dynamic range** - the ratio of a pixel's saturation level to its signal threshold. CCDs have the advantage here.
5. **Windowing** - CMOS technology has the ability to read out a portion of the image sensor allowing elevated frame rates for small regions of interest. CCDs generally have limited abilities in windowing.
6. **Shuttering** - the ability to start and stop exposure arbitrary, is superior in CCD devices. CMOS devices require extra transistors in each pixel to provide uniform (Global) shuttering and achieve similar results like CCD sensors.
7. **Biasing and clocking** - CMOS image sensors have a clear advantage in biasing and clocking, as they work on single bias voltage and clock level.
8. **Anti-blooming** - is the ability to easily reduce localized overexposure without ruining the rest of the image in the sensor. CMOS for the most part is immune to typical blooming. CCDs need higher engineering skills and additional hardware to remove blooming.

#### 4.1.5 Color Imaging with Bayer Pattern

In an area image sensor pixels are arranged in a two dimensional array (see Figure 91). Each pixel contains a light sensitive photo diode that converts the incoming light intensity into an electrical voltage. The amount of light falling into a photo diode over a period of time, defined by the exposure or integration time, determines the pixel voltage level. Based on the technology of a photo diode, each pixel is sensitive for a wide range of wavelengths, covering on silicon based sensors the whole visible as well as near infrared wavelengths. All incoming photons are accumulated to one intensity, a separation of the different wavelengths and thus color information is therefore afterwards not possible.

To build up color images, an image sensor needs the ability to extract the color information already from the incoming light. One common way for this purpose is to place a color filter array (CFA) on top of the photosensitive cells, to catch significant wavelengths individually by filtering off all others and use them to recalculate full color information for each pixel. The Bayer color filter array is the most widely used filter array on image sensors, which uses the complementary colors red, green and blue. The main advantage of this filter array is that only one image sensor is needed to separate color information of the light at one time. In a Bayer filter array there are twice as many green as there are red or blue pixels, the reason behind this is the higher sensitivity of the human eye for the color green.



#### Note

All color cameras of the Giganetix family are equipped with area image sensors with Bayer pattern.

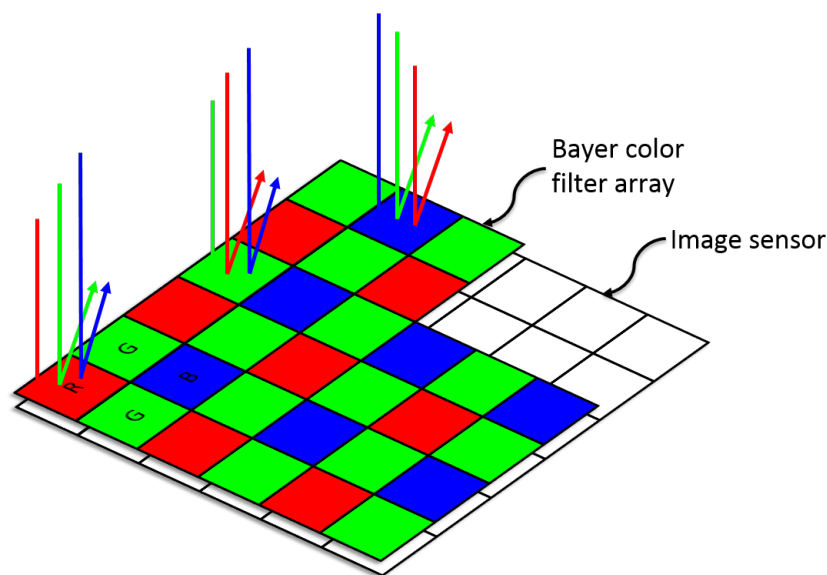


Figure 91: Bayer Color Filter Array placed on top of an area image sensor

Figure 91 illustrates a Bayer color filter array placed on top of an area image sensor:

- At a red color filter position, red light is fully transmitted, green and blue light are reflected or absorbed by the filter
- At a green color filter position, green light is fully transmitted, red and blue light are reflected or absorbed by the filter
- And at a blue color filter position, blue light is fully transmitted, red and green light are reflected or absorbed by the filter

In general the Bayer color filters are arranged in a 2-by-2 pattern where the green filter is used as twice as red or blue filter as described above. The first two pixels from top and left of the pixel array determine the name of the Bayer pattern. The Bayer pattern shown in Figure 91 is therefore called a "RG" pattern. This pattern is one of the four Bayer patterns available: GR, RG, BG and GB shown in Figure 92.

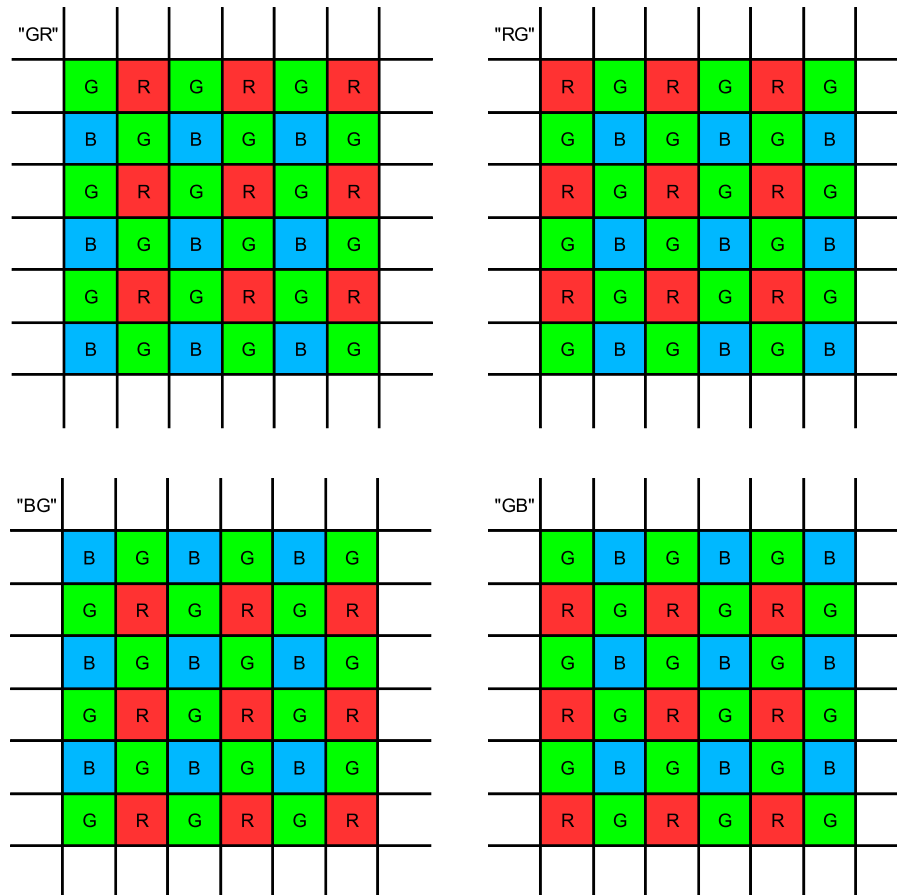


Figure 92: Bayer Color Filter Array placed on top of an area image sensor

Since each pixel accumulates only the intensity value of the red, green or blue light, there are missing information for displaying a color image. At the pixel position of a red color filter for example, the green and blue information are missing. To reproduce the full color information, various interpolation methods can be applied to calculate the missing values based on the neighbor pixels. Those interpolation methods are often called color filter array interpolation, demosaicing or debayering. For more detailed description of the debayering methods, please refer to chapter 7.2.6 - *Color Filter Array Interpolation (Demosaicing / Debayering)* in this user manual.



## 4.2 Shutter types and Frame Readout

On digital image sensors with electronic shutters, three technologies of frame shuttering are common:

- Global Shutter
- Electronic Rolling Shutter (ERS)
- Electronic Rolling Shutter with Global Reset Release (GRR)

All three technologies show up very different characteristics, which are described in the following chapter.

### 4.2.1 Global Shutter Readout

On global shutter sensors, all lines of the image sensor are exposed at the same time for an equal amount of time to incoming light. The start of exposure is defined by an incoming frame start signal (e.g. a trigger), the duration of exposure is adjusted by the user, or applied by an external signal as well.

The procedure is shown in Figure 93; the pixel in all lines are reset and started being exposed at one time, after the incoming *Frame Start* signal is received. After the *Exposure Time*, the charges of all pixel are simultaneously transferred into protected pixels on the sensor, from where they are read out line by line. The active array can usually already be exposed again while the protected pixels are still read out.

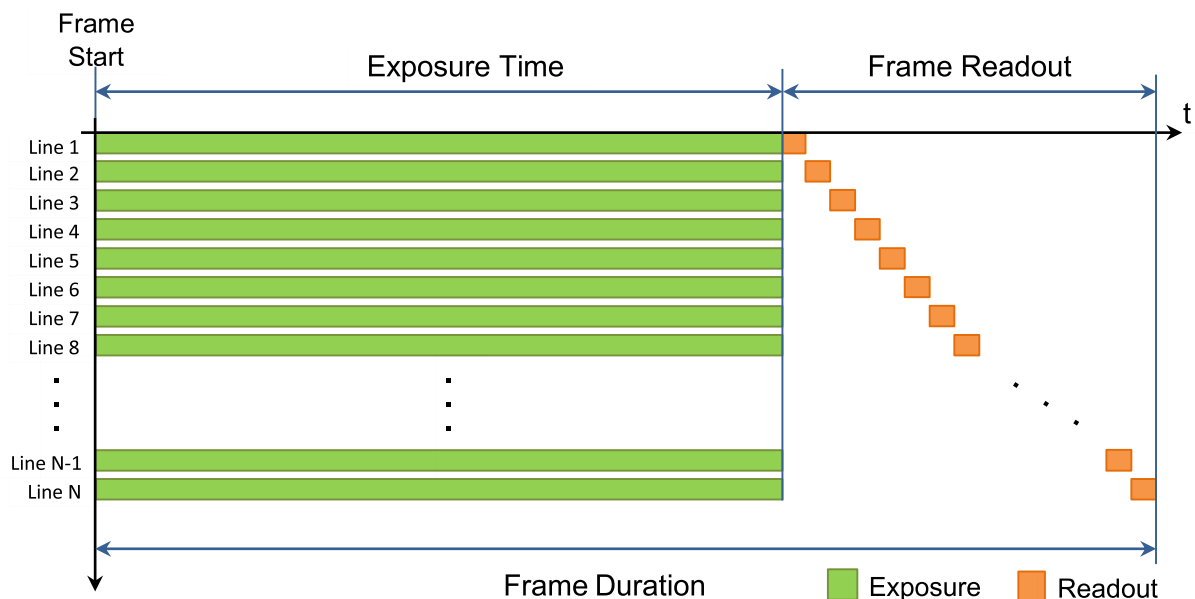


Figure 93: Global Shutter Frame Readout

Because of its characteristics to expose all lines over the same period of time, Global Shutter sensors are preferred especially for moving scenes where no additional mechanical shutter or strobe illumination is used.

To establish a global shuttering on CMOS sensor technology, further transistors need to be placed on each pixel to buffer the load of each while the sensor is read out. As this reduces the photo sensitive area of each pixel, the sensitivity of global shutter CMOS sensors tends to be smaller compared to electronic rolling shutter sensors. This is usually compensated by a micro lens above each pixel, which focuses the incoming light to the light sensitive surface.

### 4.2.2 Electronic Rolling Shutter (ERS) Readout

In contrast to the global shuttering, rolling shutter sensors start the exposure of each line not at the same moment. Each line is started to be exposed with an offset to the prior one, the exposure time of each line is defined by the user and effectively the same for all of them.

The process is shown in Figure 94; with the *Frame Start* signal, the exposure of line 1 is started. As Electronic Rolling Shutter sensors are not able to store the load of pixels in a non-photon-sensitive area, the exposure first ends with the individual pixel being read out. As the read out of all lines takes place in serial, the read out of each line is delayed by the prior ones; to keep the duration of exposure for all lines equal, the exposure start of each line is delayed about  $t_{\text{ReadRow}}$  to the prior line as well. Beside some internal timing parameters and the read out frequency,  $t_{\text{ReadRow}}$  is mainly affected by the image width. The total time for frame read out ( $t_{\text{FrameReadout}}$ ) can be calculated by multiplying  $t_{\text{ReadRow}}$  with the total count of lines in the frame.

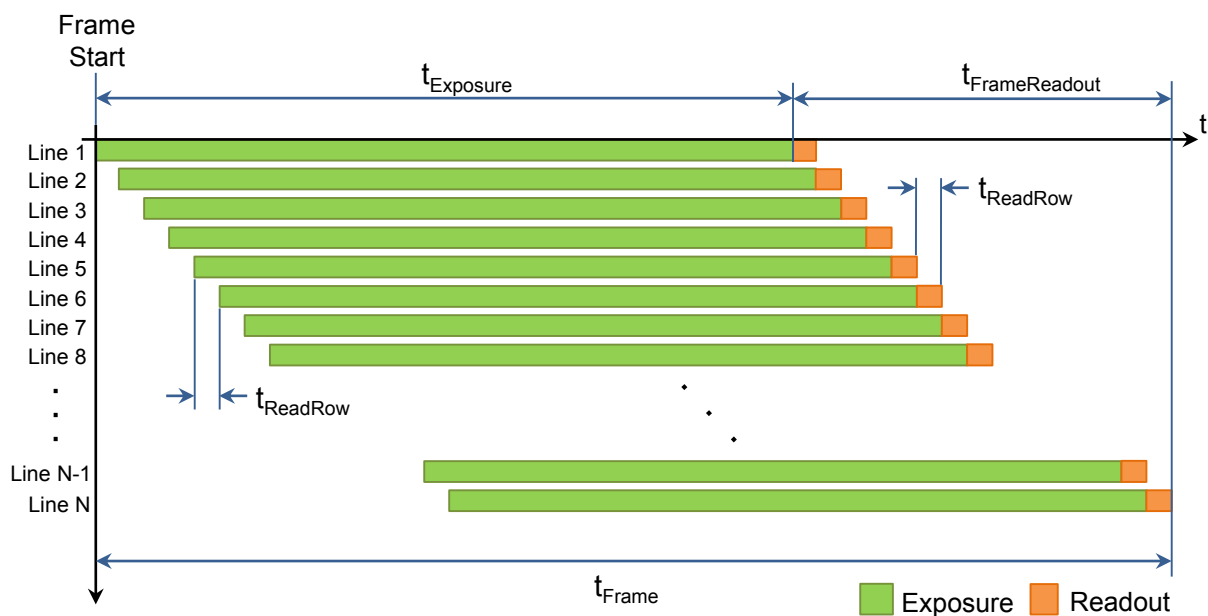


Figure 94: Electronic Rolling Shutter Frame Readout

The duration of readout per frame can be calculated with the following formula, by multiplying the time needed to read out each row with the total number of rows:

$$t_{\text{FrameReadout}} = t_{\text{ReadRow}} \times \text{ImageHeight}$$

Table 46 shows the read-out time  $t_{\text{ReadRow}}$  for the default image width of each CMOS model.

Model	$t_{\text{ReadRow}}$
GC1281	31.75 $\mu\text{s}$
GC2041	53.31 $\mu\text{s}$
GC2591	36.38 $\mu\text{s}$
GC3851	23.09 $\mu\text{s}$

Table 46: Read out time ( $t_{\text{ReadRow}}$ ) per line for CMOS Sensors (ERS)

Due to the fact that the exposure duration of each line is shifted, each of them catches a different moment of the scene, what leads to unwanted effects especially in moving scenes. This effects can be reduced or completely removed in many cases by creating a controlled illumination situation.

### Eliminating Rolling Shutter Effects

In many cases a strobe illumination or mechanical shutter can help to remove the rolling shutter effect in moving scenes by putting light onto the sensor only while all lines are within integration. Figure 95 shows this illumination window as  $t_{\text{Illumination}}$ , starting at  $t_{\text{IlluminationDelay}}$ .

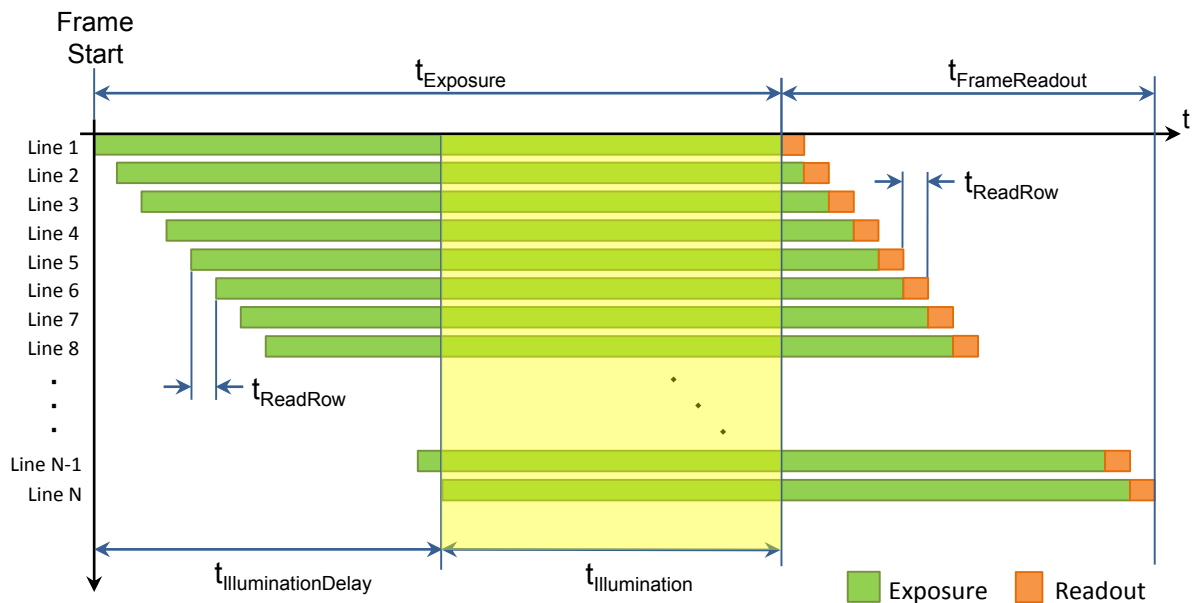


Figure 95: Electronic Rolling Shutter Frame Readout

Beyond the illumination period  $t_{\text{Illumination}}$ , ideally no light falls onto the sensor, to fully remove the rolling shutter effect. The timing of illumination or mechanical shutter can be calculated with the formulas below.

Delay of illumination / shutter open:

$$t_{\text{IlluminationDelay}} = t_{\text{ReadRow}} \times (\text{ImageHeight} - 1)$$

On time of illumination / shutter open:

$$t_{\text{Illumination}} = t_{\text{Exposure}} - (t_{\text{ReadRow}} \times (\text{ImageHeight} - 1))$$

#### 4.2.3 Global Reset Release (GRR) Readout

The Global Reset Release is a variation of the Electronic Rolling Shutter and supported by particular CMOS sensors. Like the name already indicates, all lines are reset globally at the same moment and thus also started to be exposed at the same time. As shown in Figure 96, the start of exposure of subsequent lines is not delayed like on standard rolling shutters, the readout procedure stays the same. Since the exposure duration of each line is extended about  $t_{\text{ReadRow}}$  this way to its prior, the image lightens up line by line from top to bottom.

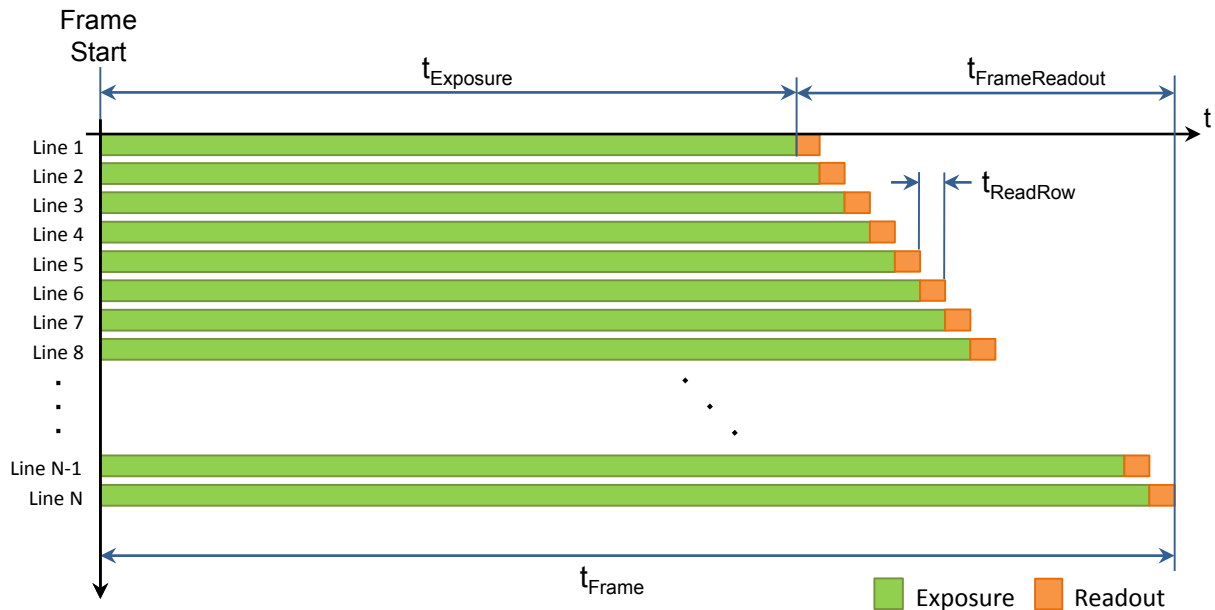


Figure 96: Global Reset Release (GRR) Frame Readout

Similar to the Electronic Rolling Shutter, the progression of brightness in the image can be reduced or even removed by a controlled illumination situation. The illumination of the sensor can in this case already be started with the sensor exposure, but must end with the exposure of Line 1, what corresponds to the overall exposure time configured in the camera.

### 4.3 Brightness and Sensor Signal Control

#### 4.3.1 Exposure / Integration Time

The brightness of an image is influenced by the amount of light that falls on the image sensor, concerning both intensity and duration. The duration of time in which the photosensitive cells of the image sensor are exposed to the incoming light is called the exposure time or the integration time. While the intensity of light depends on the light source and the lens aperture, the exposure time can be controlled by modifying parameters of the camera.

Figure 97 demonstrates two settings of camera's exposure time. The left picture is captured with an exposure time of 10000  $\mu$ s. For the right picture the exposure time is set to 22000  $\mu$ s. The brightness difference of the two images is clearly visible. Due to the nearly linear behavior of the used sensors, doubling the exposure time results in an approximately doubled pixel intensity.



Figure 97: Different exposure time settings

The exposure time for SMARTERK Vision digital cameras is configurable by the GenICam *Float* property *ExposureTime* and expressed in microseconds ( $\mu$ s). Each camera has a predefined range of values, depending on the sensor and its technology. The minimum and maximum exposure time for each camera model is shown in 2.2 - *Sensor Information and Technical Specification (All Models Separate)* and can be determined programmatically via the *IDeviceInterface* of the *gige* API.

Function	Description
bool <code>GetFloatNodeValue</code> ( "ExposureTime", double &nodeValue) const	Get value of IFloat node ExposureTime.
bool <code>SetFloatNodeValue</code> ( "ExposureTime", double nodeValue)	Set value of IFloat node ExposureTime.
bool <code>GetFloatNodeMin</code> ( "ExposureTime", double &nodeMinValue) const	Get minimum value of IFloat node ExposureTime.
bool <code>GetFloatNodeMax</code> ( "ExposureTime", double &nodeMaxValue) const	Get maximum value of IFloat node ExposureTime.

Table 47: ExposureTime - Access through API

Table 47 shows important C++ API functions in context of the exposure time, a full description of the interface and further supported languages can be found in the API documentation located in the GigEVisionSDK installation folder.



#### Note

The duration of the exposure time can affect also the maximum frame rate per second (FPS) of the camera. The exposure time in  $\mu\text{s}$  for each frame must not exceed  $\frac{10^6}{\text{TargetFPS}}$  to be able to reach the target frame rate *TargetFPS*.

The automatic modification of the camera's exposure time within user applications can be realized by using the ImageProcAPI provided by the GigEVisionSDK. For detailed description of the automatic exposure feature please refer to chapter 7.2.3 - Auto Exposure and Auto Gain.

### 4.3.2 Analog Gain and Black Level

After the charge was read out from the active pixel array it needs to be amplified according to the input levels of the analog-to-digital converter, or even higher to lighten up dark scenes without raising the exposure time or adding light.

#### Analog Gain

Figure 98 illustrates a typical image acquisition signal chain in SMARTERK Vision digital cameras. The analog voltage generated by the sensor will be passed through the Variable Gain Control where it is amplified by a factor, configurable by the camera's Gain value.

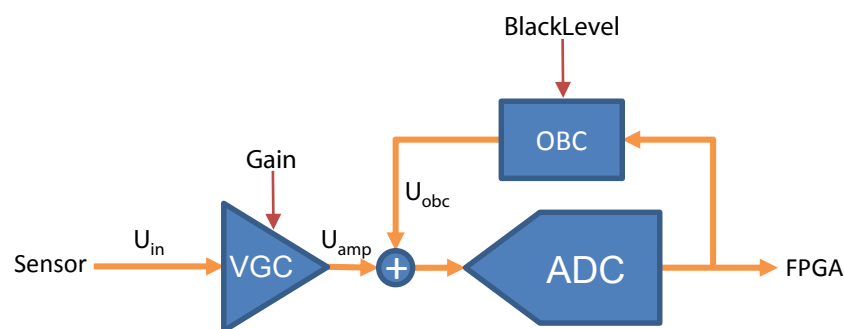


Figure 98: Typical signal chain in a CCD/CMOS image sensor

In SMARTERK Vision digital cameras gain values are expressed in decibels (dB), the analog gain defines the ratio between the output and input voltage value in a base 10 logarithmic scale:

$$\text{Gain}_{\text{dB}} = 20 \times \log_{10} \frac{U_{\text{amp}}}{U_{\text{in}}}$$

For calculating the linear amplification factor from the gain value in dB, the reverse function can be applied:

$$\frac{U_{\text{amp}}}{U_{\text{in}}} = 10^{\frac{\text{Gain}_{\text{dB}}}{20}}$$

The Gigasetix cameras provide the real value of the analog-to-digital converter to the user. As the sensor signal at saturation is usually a fraction of the level needed to generate a maximum digital value at the analog-to-digital converter, it is per default higher than 0 dB (or factor 1 in linear scale). Usual default values are between 12 dB and 15 dB, where the useful signal is fitted only to the input of the analog-to-digital converter, but not enhanced to improve e.g. the image brightness.

Gain modification is also useful for enhancing the image brightness, especially in low light condition. Increasing a gain value means increasing the intensity of each pixel, resulting in a brighter image. However, the image noise will also increase when gains are increasing. Figure 99 shows two images with different gain settings. The image on the right is captured with a gain value of 19 dB, while the image on the left is captured with a gain value of 14 dB. Like expected the right image appears brighter than the left one.





Figure 99: Captures under different gain settings

The analog gain on SMARTEK Vision digital cameras is configurable by the GenICam *Float* property *Gain* in combination with the *Enumeration* property *GainSelector*. The minimum and maximum gain values for each camera model is shown in 2.2 - Sensor Information and Technical Specification (All Models Separate) and can be determined programmatically via the *IDeviceInterface* of the *gige* API.

The following tables show important C++ API functions in context of the gain. As several cameras provide multiple gain registers giving access to the gain of individual color channels or various taps, the type of gain needs to be chosen first by the *GainSelector* property, shown in Table 48.

Function	Description
bool <code>GetEnumNodeValue</code> ( "GainSelector", double &nodeValue) const	Get value of Enumeration node <b>GainSelector</b> .
bool <code>SetEnumNodeValue</code> ( "GainSelector", double nodeValue)	Set value of Enumeration node <b>GainSelector</b> .
bool <code>GetEnumNodeValuesList</code> ( "GainSelector", StringList &nodeValueList) const	Get list of values for Enumeration node <b>GainSelector</b> .

Table 48: GainSelector - Access through API

The values for the Enumeration data type *GainSelector* can be found in Table 49, their availability depends on the camera architecture and can be requested from the camera as shown in Table 48.

GainSelector Values	Description
All, Tap1, Tap2, Tap3, Tap4	Global <b>Gain</b> (All color channels), individual per tap (multi tap sensors)
Red, Green, Blue	Individual <b>Gain</b> (per Color Channel)

Table 49: GainSelector - Values

After the appropriate gain has been selected via the *GainSelector*, its value can be get/set from the *Gain* property. Table 50 shows the most important C++ functions.



Function	Description
bool <code>GetFloatNodeValue("Gain", double &amp;nodeValue) const</code>	Get value of IFloat node <b>Gain</b> .
bool <code>SetFloatNodeValue("Gain", double nodeValue)</code>	Set value of IFloat node <b>Gain</b> .
bool <code>GetFloatNodeMin("Gain", double &amp;nodeMinValue) const</code>	Get minimum value of IFloat node <b>Gain</b> .
bool <code>GetFloatNodeMax("Gain", double &amp;nodeMaxValue) const</code>	Get maximum value of IFloat node <b>Gain</b> .

Table 50: Gain - Access through API

### Black Level

As shown in Figure 98 as well, the analog-to-digital conversion circuit includes beside the ADC an additional component with the target to remove dark current noise from the signal of each pixel. Dark current is a charge of each pixel which is generated continuously by thermal energy within the silicon lattice, even when no light enters the sensor. As its charge in the photosensitive pixels is not connected to the amount of light entering, it is no useful signal and needs to be removed before digitizing the signal, as it negatively effects the signal to noise ratio.

To help to remove the dark current noise, image sensors usually provide an array of optically shielded pixels (lines and columns, covered by a non-transmissive metal coating). Due to the coating they are at no time exposed to light and are taken as reference value for the dark current. The Optical Black Clamping (OBC) circuit, shown in Figure 98, ascertains the average digital value of the dark pixels and subtracts it from an offset, named as *clamp level*. The overall value (usually negative) is then added to the amplified signal:

$$U_{obc} = U_{amp} + \left( \text{ClampLevel} - \frac{\sum_{i=0}^n (U_{dark_i})}{n} \right)$$

The *clamp level* can be accessed by the *BlackLevel* property of the camera. It provides percentage access to the value range of the clamp level register of the analog frontend (CCD) or the sensor (CMOS) and is by default set to 0. It can be used to reduce the amount of Dark Noise subtracted from the signal or to add a user defined offset to the signal. The available clamp level ranges are shown in Table 51.

Camera Type	Clamp Level (in DN)	BlackLevel (in % of Clamp Level)
All Models (CCD Sensors)	0 to 1023	0 to 100
All Models (CMOS Sensors)	-255 to 256	0 to 100

Table 51: Range Overview of Clamp Level for CCD and CMOS Cameras



### Note

SMARTEK Vision digital cameras based on CCD technology can only apply *Analog Gain* and *BlackLevel* values to all channels (per tap) at one time. Individual gain settings (digital) can be achieved by software using the ImageProcAPI in the GigE Vision SDK.

### 4.3.3 Automatic Exposure and Gain Control

Cameras are often used in different environments and applications with changing conditions, such as scene illumination which may vary and change constantly. At certain aperture size, exposure time and gain values, the image captured by the camera can be underexposed (too dark) or overexposed (too bright), thus losing image details. As described in previous chapters the image brightness can be adjusted by changing exposure time and gain values. The SMARTEK Vision Giganetix camera series provides on-camera automatic control of exposure time and gain, thus automatically adjusting the values within defined limits, until the specified target image brightness is reached.

The operation of the automatic exposure and automatic gain algorithms can be controlled via the properties *ExposureAuto*, adjusting the exposure, and *GainAuto*, adjusting the gain of the camera. Both have three operation modes, listed in Table 52.

Value	Description
<i>Off</i>	Automatic mode is disabled; exposure and gain can be set manually via the <i>ExposureTime</i> and <i>Gain</i> properties
<i>Once</i>	Automatic Exposure/Gain is done once until the target gray value is reached, the operation mode is then set to <i>Off</i>
<i>Continuous</i>	Automatic Exposure/Gain algorithm is running continuously

Table 52: *ExposureAuto* and *GainAuto* operation modes

In case when both *ExposureAuto* and *GainAuto* are set to *Continuous* mode, the automatic control algorithm always tries to achieve the smallest gain value to keep the noise level as low as possible.

Parameters that control *Automatic Exposure* and *Automatic Gain* features are listed in the following table:

Parameter	Type	Description
<i>TargetGrayValue</i>	Integer	The average image brightness that should be reached
<i>ExposureTimeAbsLowerLimit</i>	Float	The minimum exposure time (in $\mu$ s) that is allowed to be set by the <i>Automatic Exposure</i> algorithm
<i>ExposureTimeAbsUpperLimit</i>	Float	The maximum exposure time (in $\mu$ s) that is allowed to be set by the <i>Automatic Exposure</i> algorithm
<i>GainAbsLowerLimit</i>	Float	The minimum gain value (in dB) that is allowed to be set by the <i>Automatic Gain</i> algorithm
<i>GainAbsUpperLimit</i>	Float	The maximum gain value (in dB) that is allowed to be set by the <i>Automatic Gain</i> algorithm

Table 53: *Automatic functions parameters*

For cameras that do not support on-camera automatic exposure and automatic gain control, the automatic adjustment of the camera's exposure time and gain within user application can be realized by using the ImageProcAPI provided by the GigEVisionSDK. For detailed description of the automatic exposure and gain features in the SDK please refer to chapter 7.2.3 - *Auto Exposure and Auto Gain*.

#### 4.3.4 Automatic Tap Balancing

Various models of the Giganetix series are equipped with dual tap CCD sensors. Like described in 4.1.2 - Multi-Tap CCD Sensor Readout, it is necessary to match the signal levels of all taps to receive a uniform overall image. Cameras of the Giganetix series (GC, GC-S90, GC-BL) therefore provide an *Automatic Tap Balancing* mechanism, which by default continuously adjusts the gain and black level of each tap.

The *Automatic Tap Balance* algorithm calculates the mean value of every  $n$ -th pixel residing along both sides of each tap border. Figure 100 shows this exemplarily on the basis of a 2-tap image, where relevant pixels at a *TbpVerticalStep* of 5 are tagged in green.

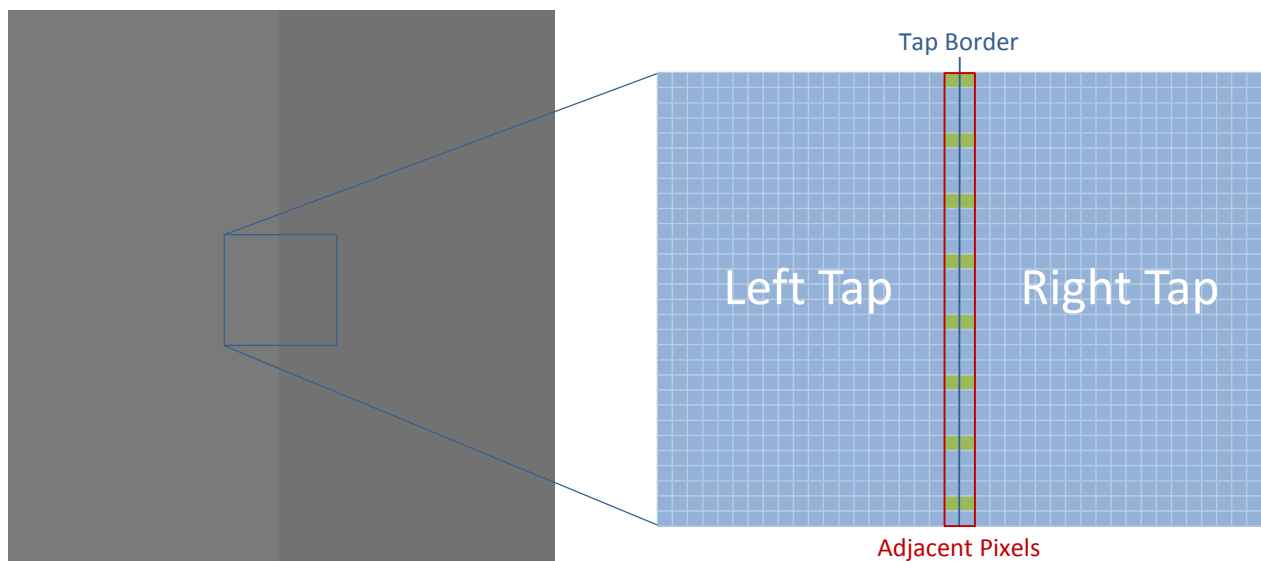


Figure 100: 2-Tap Sensor - Unbalanced Image (left) and Schematic of Adjacent Pixels

The algorithm takes into account only pixels where the difference between adjacent pixels on different taps is less than *TbpPixelDiffThreshold*. The resulting values on both taps are averaged over a count of frames defined in *TbpFramesToAvg* and compared; if the difference between the taps is larger than *TbpTapDiffThreshold* or *TbpTapDiffThresholdPercent*, the gain will be adjusted based on the size of the difference. In this process, also the difference in black level on both taps is calculated and corrected if bigger than *TbpTapDiffThresholdBL*.

An overview of the parameters is shown in Table 54 and can be accessed via the camera properties by the *GigE VisionClient* or directly via the *API*.

Parameter	Type	Description
<i>TbpVerticalStep</i>	Integer	Vertical step used in Tap Balance algorithm
<i>TbpFramesToAvg</i>	Integer	Number of frames averaged and used in Tap Balance algorithm
<i>TbpPixelDiffTreshold</i>	Float	Maximum difference between adjacent pixels on different taps to include them in calculation
<i>TbpTapDiffTreshold</i>	Float	Maximum allowed difference between taps
<i>TbpTapDiffTresholdPercent</i>	Float	Maximum allowed difference between taps in percent
<i>TbpTapDiffTresholdBLoAvg</i>	Float	Maximum allowed black level difference between taps.

Table 54: Auto Tap Balancing - Parameters



#### Note

By lowering the *TbpVerticalStep* parameter, the algorithm uses more pixels in calculation and consequently provides more precise results. On the other hand, using more pixels in calculation causes the algorithm to be slower and can lead to a decreased frame rate.



#### Note

Increasing the *TbpFramesToAvg* parameter increases the amount of frames used in averaging, localized single frame tap differences thus have smaller impact on the final result when tap difference is compared to the threshold. On the other hand, increasing the number of frames used for averaging increases the delay or reaction time between detecting the difference between taps and adjusting it.

The operation of the automatic tab balancing algorithm can be controlled via the property *GainAutoBalance*, adjusting the gain, and *BlackLevelAutoBalance*, adjusting the Black Level of the camera. Both have three operation modes, listed in Table 55.

Value	Description
<i>Off</i>	Tap balancing is disabled; taps can be balanced manually via the gain and black level properties after choosing the target tap
<i>Once</i>	Balancing of taps is done once until both taps are matched, the algorithm is then set to <i>Off</i>
<i>Continuous</i>	Tap balancing algorithm is running continuously (default)

Table 55: Parameters for *GainAutoBalance* and *BlackLevelAutoBalance*

### 4.3.5 Digital Shift

The *DigitalShift* property is part of the camera's analog controls and allows a data selection from the full bit depth of the sensor (14 Bit) to 8 Bit. As shown in Figure 101, all cameras by default use the 8 Most Significant Bits (MSB) of the 14 Bit Analog-to-Digital Converter (ADC) to create 8 Bit pixels, the Least Significant Bits (LSB) are cut off.

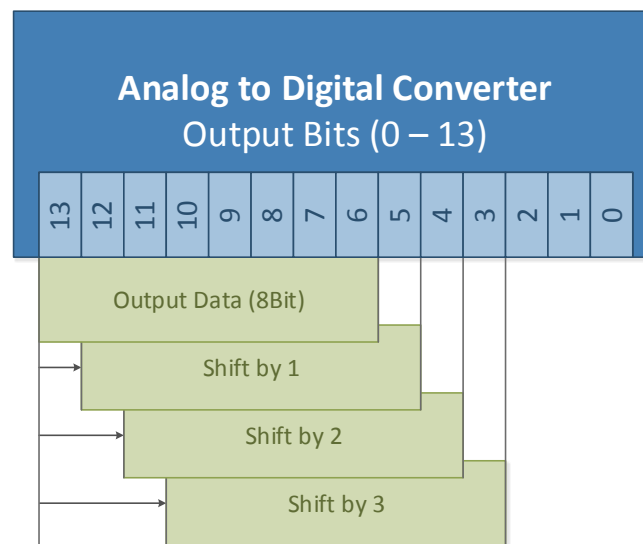


Figure 101: Digital Bit Shifting at Analog to Digital Converter

The *DigitalShift* property of the camera allows to shift the selected bits into the direction of the LSBs. As e.g. by default bits 6 to 13 are used, a *DigitalShift* value of 1 outputs bits 5 to 12 from the ADC. Table 56 shows the ADC bits outputted at each *DigitalShift* value.

DigitalShift Value	Bits at ADC
0 (default)	6 to 13
1	5 to 12
2	4 to 11
3	3 to 10
4	2 to 9
5	1 to 8
6	0 to 7

Table 56: *DigitalShift* values and used ADC bits

Shifting the significant pixels to lowers has two main effects; similar to doubling the analog amplification of the signal, the brightness in the image will double with each step. It thus enhances the maximum signal raise possible by the analog gain. Further it makes the lower bits of the ADC accessible to detect very low signals while transferring 8 bit pixels, without a further amplification of the signal.

#### 4.4 Region of Interest (ROI)

The *Region of Interest* feature allows portion of the sensor array to be read out and transmitted over transport layer. Information from pixels outside the ROI are discarded. Decreasing the ROI generally leads to increasing the maximum allowed frame rate.

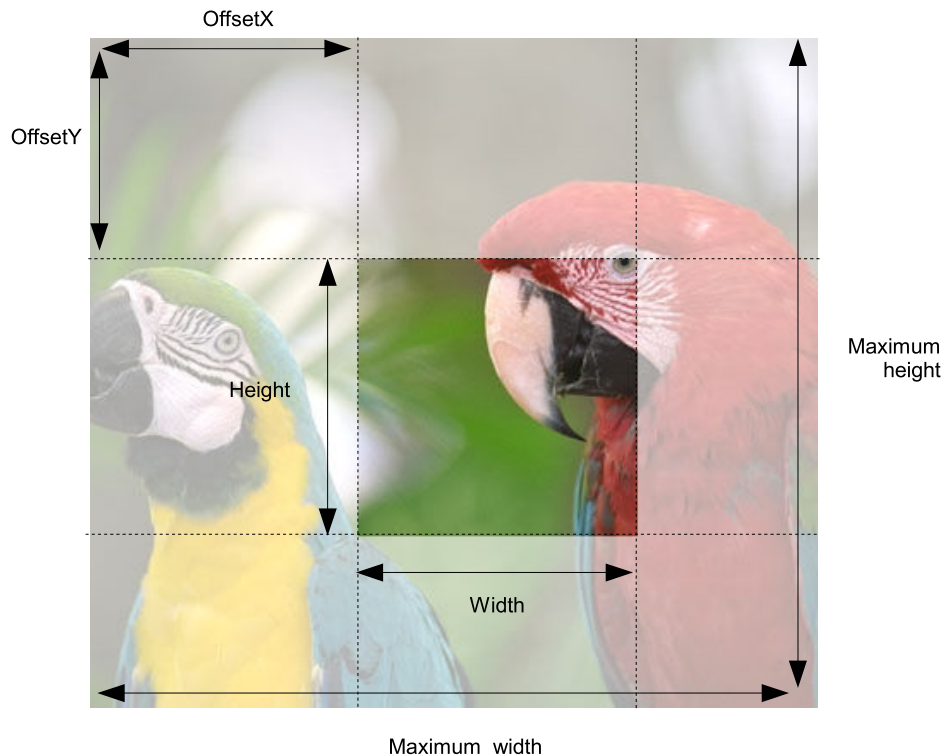


Figure 102: Region of Interest

*Region of Interest* is defined by horizontal (*OffsetX*) and vertical (*OffsetY*) offset from image origin (top-left corner) and region size in horizontal (*Width*) and vertical (*Height*) direction.

Parameter	Type	Description
<i>Width</i>	Integer	Horizontal size of the ROI image (in pixels)
<i>Height</i>	Integer	Vertical size of the ROI image (in pixels)
<i>OffsetX</i>	Integer	Horizontal offset from the origin to the ROI (in pixels)
<i>OffsetY</i>	Integer	Vertical offset from the origin to the ROI (in pixels)

Table 57: Region of Interest parameters

While the image acquisition process on the camera is active, only changes to the parameters that determine the position of the ROI are allowed (*OffsetX*, *OffsetY*). Changes to parameters that define the ROI size (*Width*, *Height*) are not allowed. Changes to parameters while the image acquisition process is active is also called "on-the-fly" changes.

#### 4.4.1 Multiple Regions of Interest

If a camera supports *Multiple Regions of Interest*, the parameters *RegionSelector*, *RegionMode* and *RegionDestination* can be used to select and control each region individually.

Parameter	Type	Description
<i>RegionSelector</i>	Enumeration	Selects the Region of Interest to control
<i>RegionMode</i>	Boolean	Enables or Disables the selected region
<i>RegionDestination</i>	Enumeration	Controls the destination stream of the selected region

Table 58: Multiple Regions of Interest parameters

Each ROI defines one horizontal and one vertical stripe in the full resolution image as shown in Figure 103. The stripes that are overlapping or are located next to each other are merged into one stripe. For example, horizontal stripe belonging to Region0 (ROI1) and horizontal stripe belonging to Region1 (ROI2) are merged into HStripe1. The resulting *Multiple ROI* image will contain all image areas where horizontal and vertical stripes overlap.

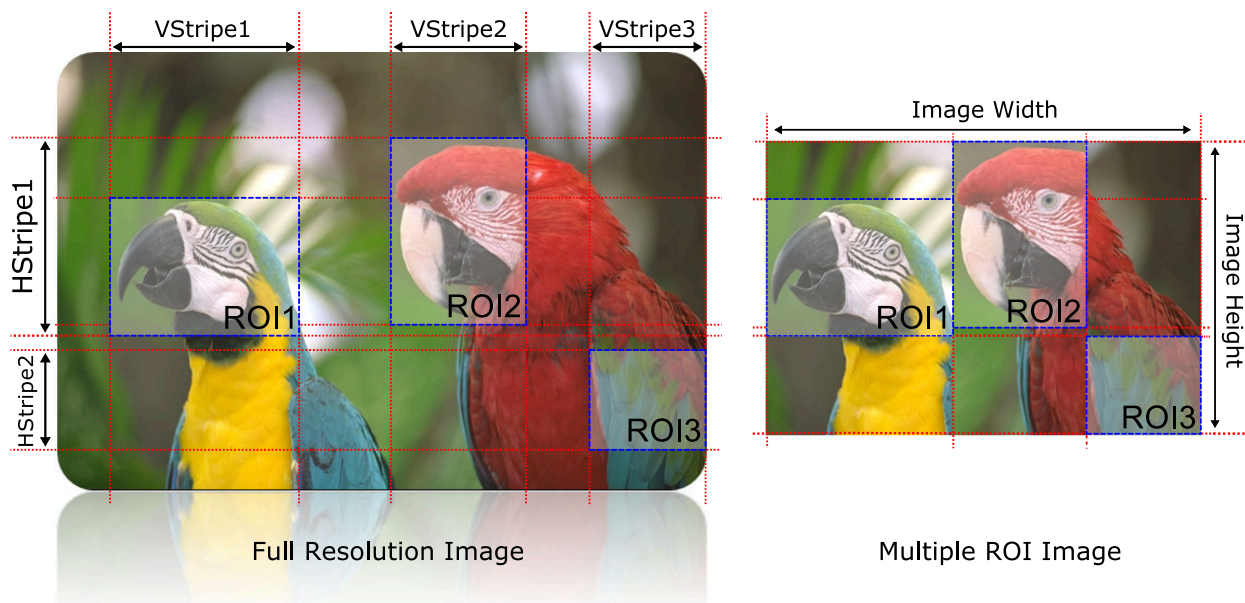


Figure 103: Multiple Regions of Interest

The width of the resulting image is equal to the sum of widths of all vertical stripes:

$$\text{ImageWidth} = \sum \text{VStripe.width}$$

The height of the resulting image is equal to the sum of widths of all horizontal stripes:

$$\text{ImageHeight} = \sum \text{HStripe.width}$$



The parameters of the example shown in Figure 103 are listed in Table 59 below. It shows the definition of all three ROIs.

	ROI 1	ROI 2	ROI 3
<i>RegionSelector</i>	"Region0"	"Region1"	"Region2"
<i>RegionMode[RegionSelector]</i>	"On"	"On"	"On"
<i>RegionDestination[RegionSelector]</i>	"Stream0"	"Stream0"	"Stream0"
<i>Width[RegionSelector]</i>	576	400	360
<i>Height[RegionSelector]</i>	392	520	288
<i>OffsetX[RegionSelector]</i>	160	1000	1576
<i>OffsetY[RegionSelector]</i>	408	232	824

Table 59: Multiple Regions of Interest example

When more than one *Region of Interest* is enabled, "on-the-fly" changes are not allowed even for parameters that determine the position of the ROI (*OffsetX*, *OffsetY*) as they might influence the horizontal and vertical stripes forming the resulting image. "On-the-fly" change of the parameter *RegionMode* is also not allowed as it results in enabling or disabling of a *Region of Interest*.



#### 4.4.2 Region of Interest Centering

When parameters CenterX and CenterY are enabled, camera automatically calculates and sets the horizontal and vertical offsets positioning the ROI in the center of the image. Parameters OffsetX/OffsetY are unavailable when CenterX/CenterY are enabled.

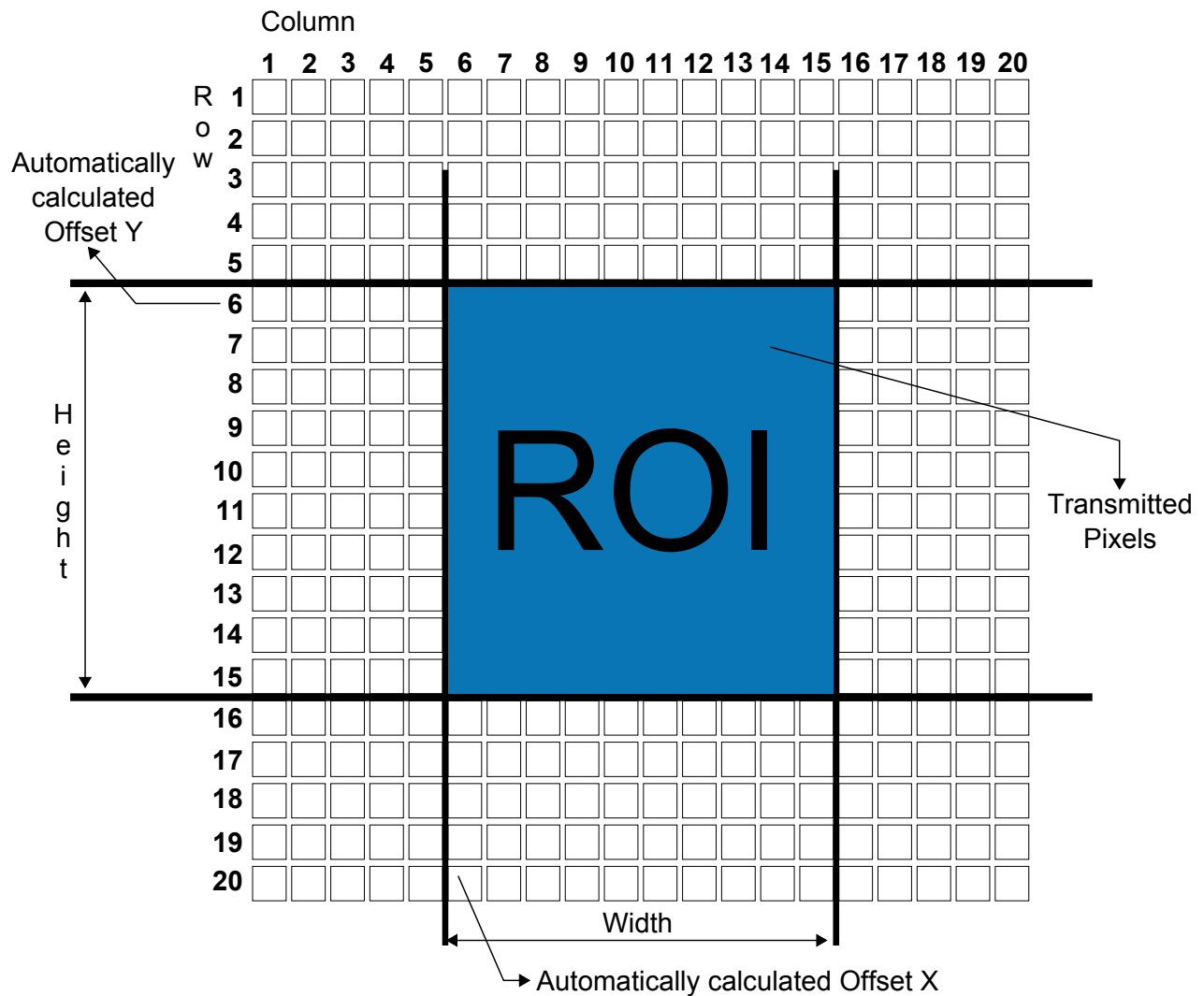


Figure 104: Center X & Y



#### Note

On cameras equipped with 4-Tap CCD image sensors *ROI Centering* feature will generally result in higher achievable frame rate compared to arbitrarily set ROI position.

## 4.5 Acquisition Control

The following section is about controlling the image acquisition of SMARTEK Vision digital cameras. It contains a detailed description about the different acquisition modes, how to control external triggering and how the image acquisition rate can be limited. Table 60 gives a brief overview about all features that are available to control the image acquisition in the cameras.

Image Acquisition Features	Short description
<i>AcquisitionMode</i>	Defines the number of frames to be captured. Three options are available: <ul style="list-style-type: none"> <li>• Continuous</li> <li>• SingleFrame</li> <li>• MultiFrame</li> </ul>
<i>AcquisitionStart</i>	Start acquisition
<i>AcquisitionStop</i>	Stop acquisition
<i>AcquisitionAbort</i>	Abort acquisition
<i>AcquisitionFrameCount</i>	Number of frames to acquire in MultiFrame acquisition mode
<i>AcquisitionBurstFrameCount</i>	Number of frames to acquire for each FrameBurstStart trigger
<i>AcquisitionFrameRate</i>	Controls the acquisition rate (in Hz) at which the frames are captured
Trigger Features	Short description
<i>TriggerMode</i>	Enable/Disable the trigger mode. Two options are available: <ul style="list-style-type: none"> <li>• On</li> <li>• Off</li> </ul>
<i>TriggerSoftware</i>	Generate a software trigger
<i>TriggerSource</i>	Select the source that fires a trigger signal: <ul style="list-style-type: none"> <li>• Line1: Physical Input Line 1</li> <li>• Line2: Physical Input Line 2</li> <li>• Software</li> </ul>
<i>TriggerActivation</i>	Define the clock edge of the input signal for activate triggering <ul style="list-style-type: none"> <li>• Rising Edge</li> <li>• Falling Edge</li> </ul>
<i>TriggerDelay</i>	Specify the delay in microseconds ( $\mu$ s) to incoming trigger signals.

Table 60: Camera features for image acquisition control

In the following chapter the image acquisition is divided into two general types - the *Free Run* operation, where the camera streams images as fast as possible and the *Triggered* operation, where the camera waits for a further signal by an external source to start the acquisition of a count of images.

### 4.5.1 Free Run Operation

In Free Run mode the camera starts the acquisition of images as soon as the *AcquisitionStart* command was received by the device. Images are streamed with a by parameters fixed frequency, which by default corresponds to the maximum of the camera. By the Acquisition Modes it is furthermore possible to define how many images are acquired and streamed after receiving the *AcquisitionStart* command, until the acquisition is stopped again.

#### 4.5.1.1 Acquisition Modes

The *AcquisitionMode* property controls the acquisition mode of the device. It defines the number of frames captured during the acquisition and the way the acquisition stops. It can take any of the values shown in Table 61.

Value	Description
<i>Continuous</i>	Frames are captured continuously until stopped by the <i>AcquisitionStop</i> command
<i>SingleFrame</i>	The camera captures only one frame and stops the acquisition
<i>MultiFrame</i>	The camera captures a specific number of frames set by the <i>AcquisitionFrameCount</i> property and stops the acquisition

Table 61: *AcquisitionMode* values

In order for the camera to run in free run, in which the camera acquires and transfers images at maximum configured frame rate, the *TriggerMode* properties for all *TriggerSelector* need to be set to *Off*.

#### 4.5.1.2 Acquisition Frame Rate

The *AcquisitionFrameRate* property is a feature which limits the frequency at which images are captured by the camera. Using the *AcquisitionFrameRate* feature it is possible to decrease the number of frames the camera acquires and transmits in free run mode, which consequently lowers the Ethernet bandwidth needed by the camera.

This feature is useful in situations where the Ethernet bandwidth is limited, like in applications where several cameras acquire images using one single Gigabit Ethernet link.

Setting the *AcquisitionFrameRate* property to zero effectively disables the feature, allowing the camera to acquire and transfer images at maximum frame rate.

### 4.5.2 Triggered Operation

The trigger operation mode enables the user to precisely synchronize the camera with a further device or software application. The *AcquisitionStart* command sets the sensor in stand-by, waiting for a trigger signal. In contrary to the free run mode, the sensor is already initialized when a trigger initiates the integration, which is thus started with a minimum of latency.

Similar to the free run mode, also the trigger mode allows several kinds of operation. The selection of a trigger mode is done with the *TriggerSelector* property. Each trigger mode has its own parameters, all of them can be active at the same time. The camera is in trigger operation mode as soon as one of the modes selectable by the *TriggerSelector* is enabled via its *TriggerMode* property.

#### 4.5.2.1 Trigger Selector

The *TriggerSelector* property provides access to the settings of the different trigger operating modes supported by the camera, shown in Table 62.

Value	Description
<i>AcquisitionStart</i>	Starts the acquisition like configured in <i>AcquisitionMode</i>
<i>FrameStart</i>	Starts the acquisition of a single frame
<i>FrameBurstStart</i>	Starts the acquisition of multiple frames; the count of frames is defined in the <i>AcquisitionBurstFrameCount</i> property.

Table 62: Supported Types of Triggers

All available modes can be configured separately and are valid in parallel, allowing to assign different behavior to each trigger source.

#### 4.5.2.2 Trigger Mode

The property *TriggerMode* enables and disables the triggered operation of the camera. As soon as it is enabled for at least one *TriggerSelector* property, the camera's sensor falls in stand-by mode where it waits for an external signal defined in the *TriggerSource* property. The *TriggerMode* property is individually available for each *TriggerSelector* and can take one of the values shown in Table 63.

Value	Description
<i>On</i>	Enables trigger operation for the current <i>TriggerSelector</i>
<i>Off</i>	Disables trigger operation for the current <i>TriggerSelector</i>
<i>FrameBurstStart</i>	Starts the acquisition of multiple frames; the count of frames is defined in the <i>AcquisitionBurstFrameCount</i> property.

Table 63: Trigger Mode

While all *TriggerMode* properties are set to *Off* and the *AcquisitionMode* property to *Continuous*, the camera acquires continuously images.

### 4.5.2.3 Trigger Source

The *TriggerSource* property specifies the source which is used to initiate the trigger signal. An internal signal or one of the physical input lines can be selected as the trigger source. The selected *TriggerSelector* must have its *TriggerMode* set to On, *TriggerSource* can take any of the values shown in Table 64.

Value	Description
<i>Line1</i>	Uses the physical Input Line1 as trigger source
<i>Line2</i>	Uses the physical Input Line2 as trigger source
<i>Software</i>	Uses a Software Trigger as trigger source; the <i>TriggerSoftware</i> command can be send to the camera via the API

Table 64: Trigger Sources

### 4.5.2.4 Trigger Activation

For the external trigger signals applied on the physical input lines of the camera available in the *TriggerSource* property, it is possible to define the kind of edge which initiates a trigger. The *TriggerActivation* property offers two values described in Table 65.

Value	Description
<i>RisingEdge</i>	Trigger initiated with Rising Edge
<i>FallingEdge</i>	Trigger initiated with Falling Edge

Table 65: Trigger Activation modes

Figure 105 shows the exposure period of the sensor triggered by a rising edge, while Figure 106 shows the exposure period of the sensor triggered by a Falling Edge.

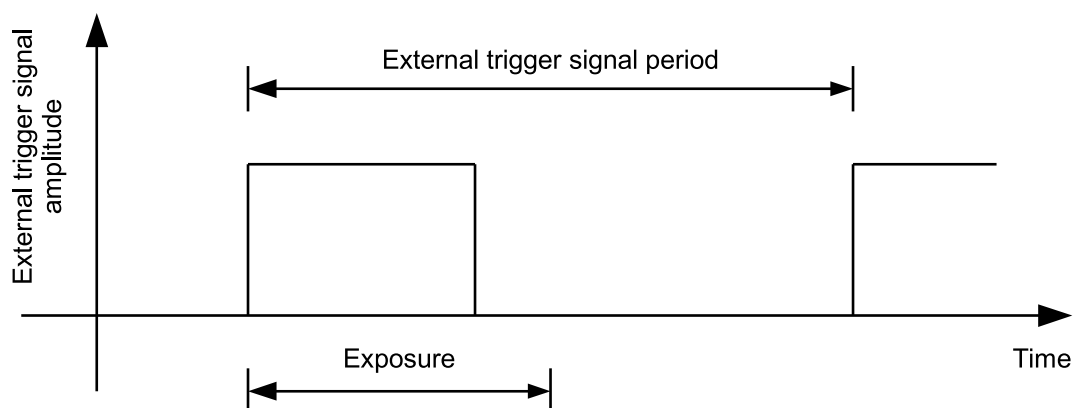


Figure 105: Exposure with a rising edge of the trigger

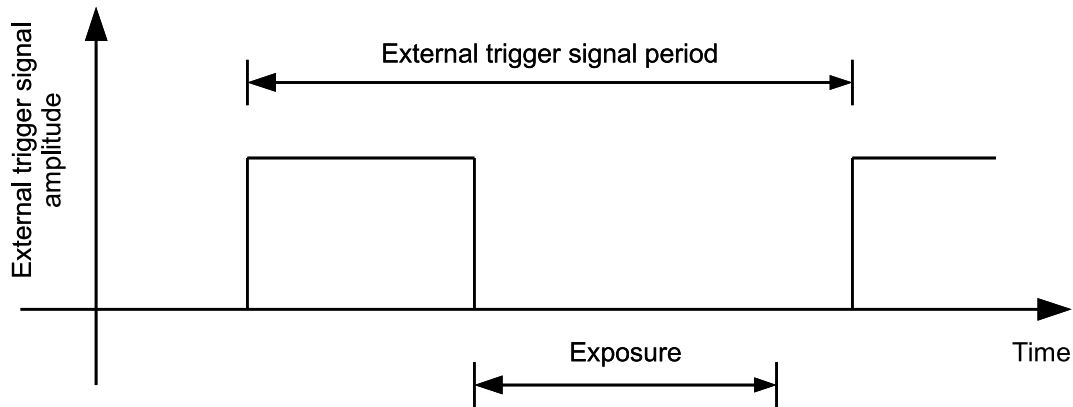


Figure 106: Exposure with a falling edge of the trigger

#### 4.5.2.5 Trigger Delay

The *TriggerDelay* property defines a period of time for which an incoming trigger signal is delayed, until it is internally used to trigger the image sensor. The trigger delay is expressed in  $\mu\text{s}$  and can be set manually by software. For external signals, entering a physical input line, a general latency of  $2\mu\text{s}$  is added by the input line circuitry of the camera. Together with the *TriggerDelay* it represents the signal latency in the camera until the image sensor is triggered.

A further description of the complete trigger process can be found in 4.6.1 - *Input Lines*.

## 4.6 Digital Input / Output Control

The digital inputs and outputs of the Gigasetix series can be used to synchronize the camera with other devices and cameras. The camera can be triggered on a rising or falling edge of the input trigger signal, or trigger other devices on configurable events. The physical interface is provided via the *General Purpose Input and Output* (GPIO) connector, described in 2.3.2 - Power and I/O-Interface.

Each physical line is configured separately and can be selected by the *LineSelector* property. The property *LineMode* contains if the currently selected line is an *Input* or *Output*. Table 66 describes all available configuration properties.

Property	Type	Description
<i>LineSelector</i>	Enumeration	Select the line for configuration; all further properties contain the values based on the selected line. Values are: <ul style="list-style-type: none"> <li>• Line1</li> <li>• Line2</li> <li>• ...</li> </ul>
<i>LineMode</i>	Enumeration	Contains if the currently selected line is an Input or Output
<i>LineStatus</i>	Boolean	Current status of the selected line
<i>LineSource</i>	Enumeration	Source event driving the output line
<i>LineFormat</i>	Enumeration	Internal (electrical) circuit of the line
<i>LineDebouncerTime</i>	Float	Define the debouncer time of input lines (in $\mu\text{s}$ )

Table 66: Trigger Sources

### 4.6.1 Input Lines

The camera's input lines can be used to trigger an acquisition of the camera on external events. The assignment of the physical input lines as a frame or acquisition trigger is described in Chapter 4.5.2 - *Triggered Operation*. Figure 107 shows the partial process of a complete image acquisition. The incoming electrical signal of the external source is filtered by the *LineDebouncer* and raises the internal trigger signal (if valid). The internal circuit adds a fixed trigger latency stated in the camera's specification (usually  $\sim 2\mu\text{s}$ ) to which the user defined *TriggerDelay* is added until the exposure is started.

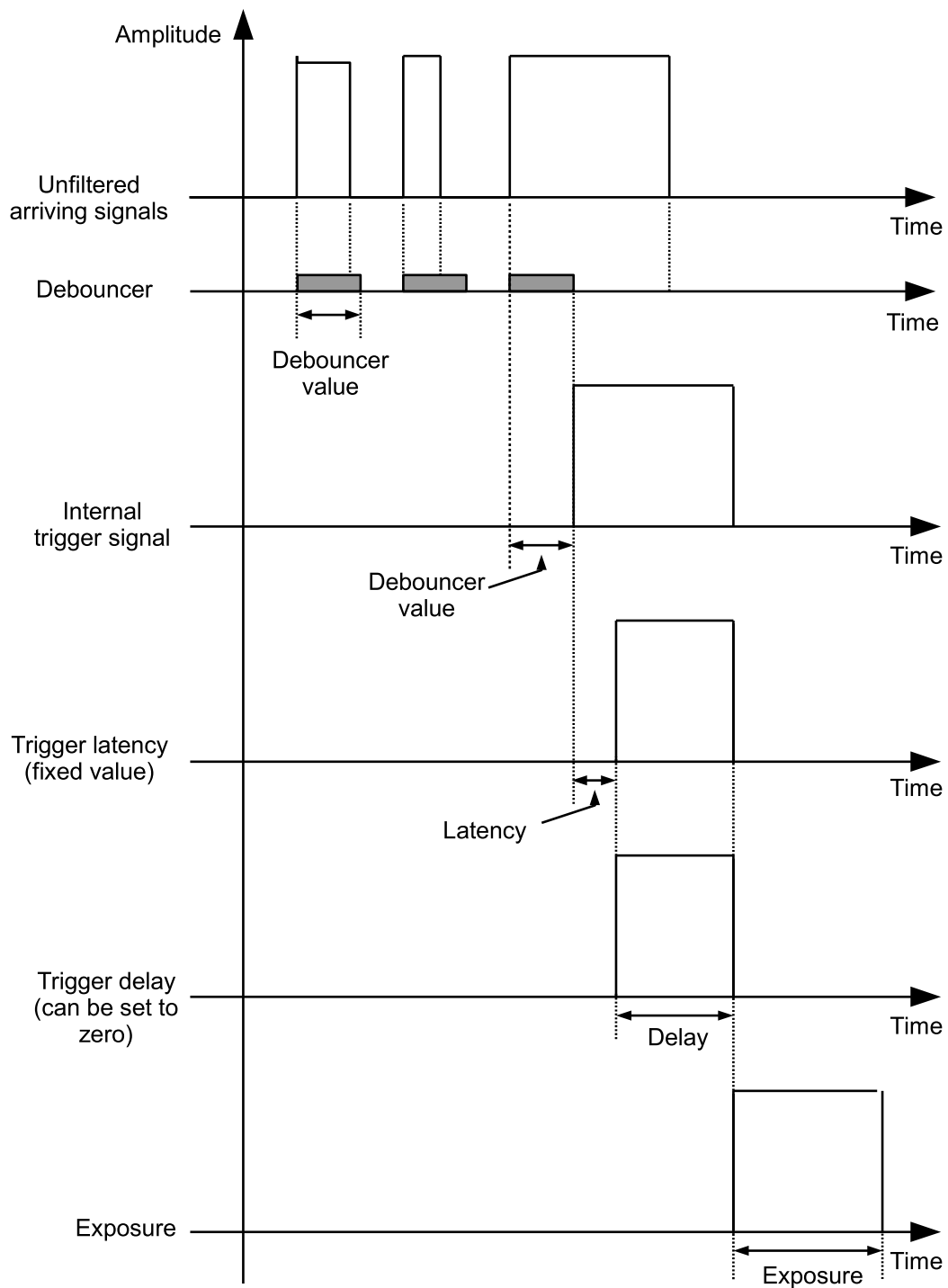


Figure 107: Partial process of image acquisition



### Line Debouncer

The *LineDebouncer* property defines the minimum time interval that an input signal must remain active in order to be recognized as valid trigger. The line debouncer is used to prevent possible unwanted trigger events as it eliminates short noises that could easily be interpreted as trigger signal. The function of the trigger debouncer is shown in Figure 108; two glitches are ignored by the debouncer because the width of these signals is shorter than the debouncer time value. The third signal is accepted as a valid trigger signal as its width is longer than the debouncer time limit. The *LineDebouncerTime* feature is used to set the line debouncer time expressed in  $\mu\text{s}$ .

The line debouncer time effectively increases delay time between external trigger signal and internal trigger signal used to start the exposure, so it should be set large enough to filter unwanted glitches that could trigger the camera, but small enough to keep the delay as small as possible.

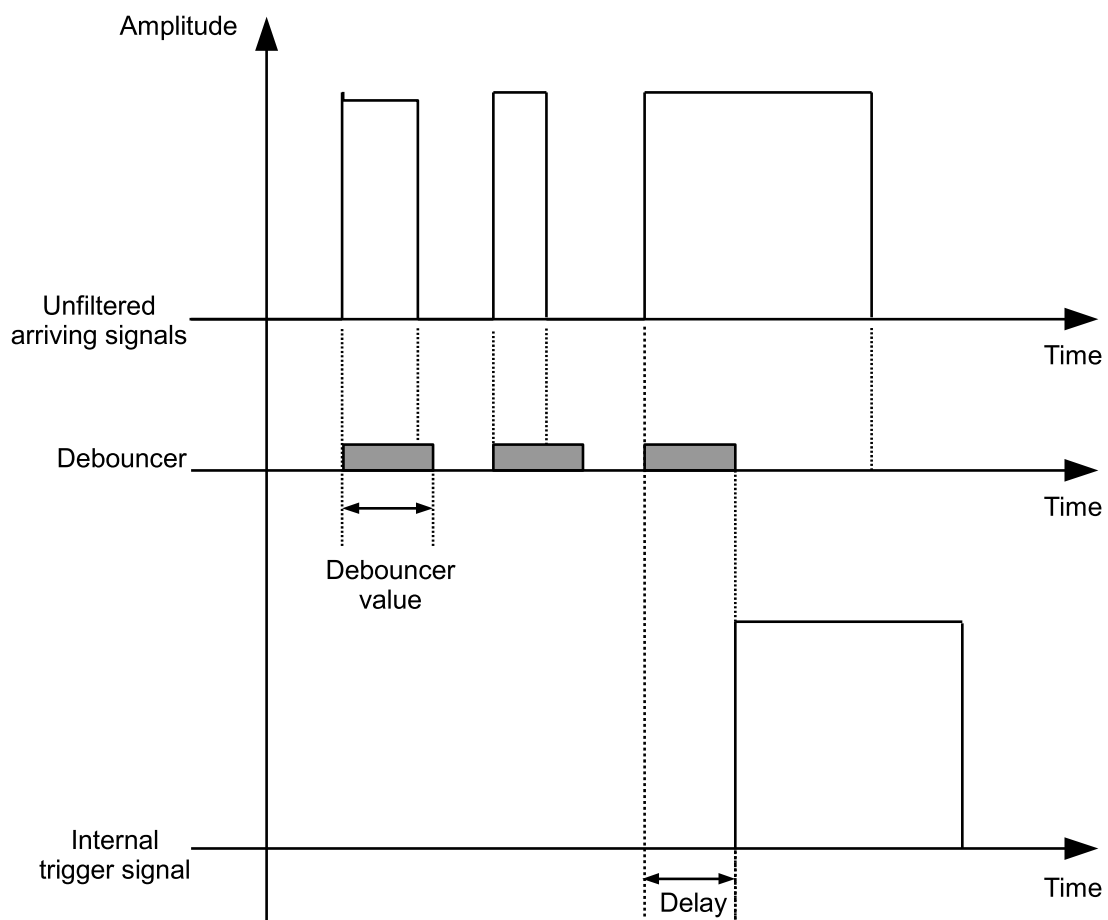


Figure 108: Line debouncer function

### 4.6.2 Output Lines

The physical output lines are usually used to synchronize the camera to external devices in situations where it makes sense that the camera is the master. Usual use cases are for example the synchronization of external illuminations to the sensor exposure or to drive mechanical actuators by the software application.

A full list of possible events assignable by the *LineSource* property to each output line is shown in Table 67.

Value	Description
<i>UserOutput1 / UserOutput2</i>	Set to High / Low by the software application with the <i>UserOutputSelector / UserOutputValue</i> properties
<i>AcquisitionTriggerWait</i>	High while the camera waits for a trigger for one or more frames
<i>AcquisitionActive</i>	High while camera acquires one or more frames
<i>FrameTriggerWait</i>	High while camera waits for frame trigger
<i>FrameActive</i>	High while camera captures a single frame

Table 67: Trigger Sources

While nearly all events available for the *LineSource* property are raised by the current acquisition state of the camera, the *UserOutput* can be defined by the user and thus be used to set a physical output line manually. The value of this property can be accessed after selecting the appropriate value by the *UserOutputSelector*, according to Table 68.

Value	Type	Description
<i>UserOutputSelector</i>	Enumeration	Select between <i>UserOutput1</i> and <i>UserOutput2</i>
<i>UserOutputValue</i>	Boolean	Value of the selected <i>UserOutput</i>

Table 68: User Outputs

## 5 Image Transmission over Gigabit Ethernet

The network interface of SMARTEK Vision digital cameras is designed to be fully compatible with the GigE Vision standard. The following section describes features of the data interface of the Giganetix series as well as the SMARTEK Vision GigEVision Filter Driver. Further, several optimization settings of the network driver are introduced and the pixel structure of the image data is elaborated.

### 5.1 Smartek GigE Vision Filter Driver

Ethernet packets can come from various sources and do not have to be in order. This makes it necessary that each data segment is usually routed through the whole network stack of the operating system until it is delivered to the target application. The CPU load can especially at high data rates be significantly affected by this process as well as the latency of the image stream. The UDP protocol, used for the transmission with low latencies, is furthermore not able by default to handle packet loss - packet collisions lead inevitably to a loss of image data.

To optimize the data transmission and add features like an optimized packet resend mechanism, SMARTEK Vision provides its own GigE Vision filter driver. The filter driver separates streaming video packets from the rest of network data already before the network stack of the operating system gets access to them and forwards them directly to the application. It is designed to be compatible with the most network adapter cards and significantly lowers the CPU load needed to service the network traffic between the PC and the camera(s).

#### 5.1.1 UDP Packet Resend Mechanism

The *Packet Resend* mechanism is a driver feature that, when enabled, tries to regain packets that have been lost during transmission. It checks the order of the incoming packets and detects if one or even a group of packets is missing in a stream. Depending on the parameters settings the driver will then send one or several resend requests to the camera which resends the appropriate packets. Table 69 shows the basic driver parameters available.

Parameter	Type	Description
<i>SetParametersToDefault</i>	CMD	Resets all parameters of the Packet Resend mechanism to the default values
<i>MaxImageSize</i>	Integer	Maximum image size that the Packet Resend mechanism will handle
<i>EnablePacketResend</i>	Boolean	Enables / Disables the Packet Resend mechanism
<i>AcceptIncompletImage</i>	Boolean	Enables / Disables the acceptance of images where payload packets are missing
<i>LineFormat</i>	Enumeration	Internal (electrical) circuit of the line

Table 69: Packet Resend Parameters (1/2)

Table 70 lists further parameters, allowing a detailed configuration of the *Packet Resend* mechanism. All this parameters mainly affect the performance and the robustness of the packet resending, changes should only be done carefully.

Parameter	Type	Description
<i>PacketResendTimeout</i>	Integer	The elapsed time (in ms) before the first resend request for a missing packet is sent to the camera. The default value is 0 ms, meaning the request for a missing packet will be sent instantly. This parameter applies only once to each missing packet after the packet was detected as missing.
<i>PacketResendResponseTimeout</i>	Integer	Represents how long (ms) the Packet Resend Mechanism will wait for response after sending a resend request, until another resend request is sent.
<i>MaxResendPacketRetry</i>	Integer	Represents the maximum number of resend requests sent for a missing packet.
<i>MaxMissingPacketWaiting</i>	Integer	Maximum time (in ms) the missing packet is waited for. When this time expires, there will be no more resend requests sent to camera even if the driver did not send all resend request specified with MaxResendPacketRetry and the packet will be considered as lost.
<i>MaxNextPacketWaiting</i>	Integer	Maximum time (ms) that the resend mechanism will wait for the next packet. If this time expires and there are still retries left, the resend request is sent again.
<i>MaxMissingPacketsCount</i>	Integer	Maximum number of missing packets in one frame. If the frame has more missing packets then this value it will be dropped.
<i>MaxNewImagesPending</i>	Integer	Maximum amount of new images pending in the buffer. Current image is dropped if this amount is exceeded.
<i>MaxNewPacketsPending</i>	Integer	Maximum amount of incoming payload packets pending in the buffer. Current frame is dropped if this amount is exceeded.
<i>MaxIncompletePackets</i>	Integer	Maximum amount of missing payload packets from a block of the image to be accepted.

Table 70: Packet Resend Parameters (2/2)



### Note

In healthy networks it is not necessary to recover more than a small number of packets per hundreds of transmitted images. Should the amount of packet resends rise to a unnatural height, check the correctness of the physical network setup (cabling, switches) and the network optimization settings located in chapter 5.5 - *Network Interface Optimization*.

### Example 1

Figure 109 illustrates the packet resend mechanism with the following assumptions:

- Packet 1007 is missing within the stream of packets and has not been recovered
- *MaxResendPacketRetry* parameter is set to 2

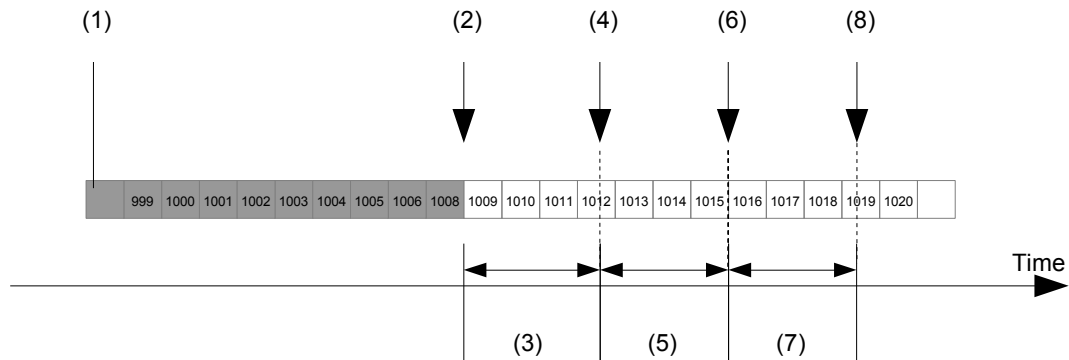


Figure 109: Packet Resend example when *MaxResendPacketRetry* value has exceeded

Corresponding to Figure 109, the workflow of the Packet Resend mechanism would look like described below:

1. Stream of packets. Gray indicates packets that have been checked by the driver, white packets have not yet been checked.
2. As packet 1008 is received, packet 1007 is detected as missing.
3. Interval defined by the *PacketResendTimeout* parameter.
4. The *PacketResendTimeout* is expired, the first resend request for packet 1007 is sent to the camera. The camera does not respond with a resend.
5. Interval defined by the *PacketResendResponseTimeout* parameter.
6. The *PacketResendResponseTimeout* expires and second resend request for packet 1007 is sent to the camera. The camera does not respond with a resend.
7. Interval defined by the *PacketResendResponseTimeout* parameter.
8. As the maximum number of resend requests has been sent (*MaxResendPacketRetry*) and the last *PacketResendResponseTimeout* has expired, packet 1007 is now considered as lost.

If a group of packets is missing (for example 1000, 1001, 1002 and 1003), only one resend request will be sent covering all connected packets.

### Example 2

Figure 110 illustrates the packet resend mechanism with the following assumptions:

- Packet 1007 is missing within the stream of packets and has not been recovered.
- *MaxResendPacketRetry* is set to 2.
- *MaxMissingPacketWaiting* is set to a value that expires before second resend request is sent.

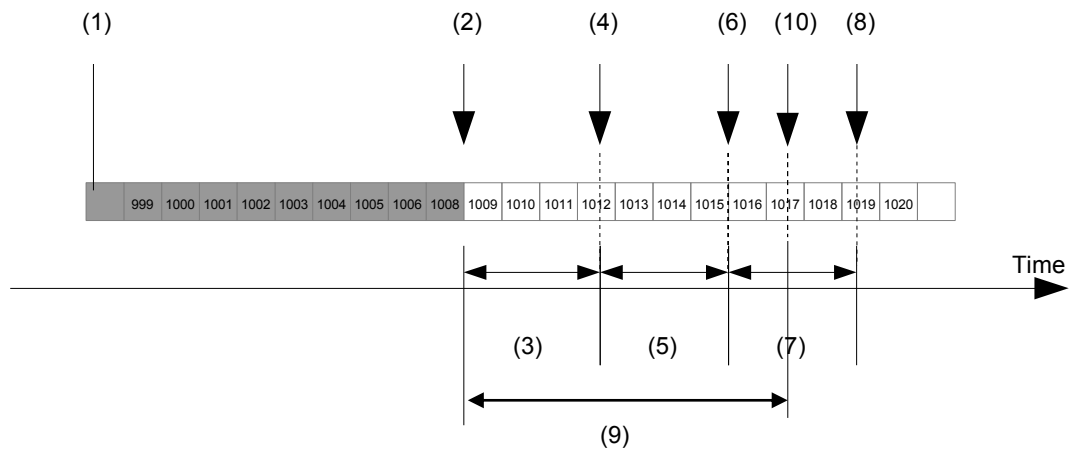


Figure 110: Packet Resend Mechanism example *MaxMissingPacketWaiting* value has exceeded

Additionally to the description in Example 2, the workflow of the Packet Resend mechanism would be enhanced by the following definitions:

1. Interval defined by *MaxMissingPacketWaiting* parameter.
2. As the *MaxMissingPacketWaiting* time has expired, missing packet is considered as lost.

### 5.1.1.1 API Statistics

The *Packet Resent* statistics in the *API* can be accessed by the parameters described in Table 71.

Parameter	Type	Description
<i>ResetAll</i>	CMD	Resets all current statistics
<i>MissingPackets</i>	Integer	Count of missing packets
<i>PacketResendsAmount</i>	Integer	Count of resend requests
<i>LostPackets</i>	Integer	Count of packets that were declared as lost
<i>LostImages</i>	Integer	Count of images that were declared as lost
<i>IgnoredPackets</i>	Integer	Count of packets that are ignored, usually packets that are already analyzed or packets that do not belong to this stream
<i>IncompleteImages</i>	Integer	Count of processed incomplete images
<i>AllPackets</i>	Integer	Count of all received packets
<i>UnknownDevice</i>	Integer	Count of all unknown devices
<i>LeaderPackets</i>	Integer	Count of all leader packets that are processed
<i>PayloadPackets</i>	Integer	Count of all payload packets that are processed
<i>TrailerPackets</i>	Integer	Count of all trailer packets that are processed
<i>UnknownPackets</i>	Integer	Count off all packets that have wrong packet format and were discarded

Table 71: *Packet Resent* statistic parameters

### 5.1.1.2 Device Packet Statistics

The Packet Resent statistics of the device / camera can be accessed by the parameters described in Table 72.

Parameter	Type	Description
<i>ResetAll</i>	CMD	Resets all current statistics
<i>MissingPackets</i>	Integer	Count of missing packets
<i>PacketResendsAmount</i>	Integer	Count of resend requests
<i>LostPackets</i>	Integer	Count of packets that were declared as lost
<i>LostImages</i>	Integer	Count of images that were declared as lost
<i>IgnoredPackets</i>	Integer	Count of packets that are ignored, usually packets that are already analyzed or packets that do not belong to this stream
<i>IncompleteImages</i>	Integer	Count of processed incomplete images

Table 72: Packet Resent statistic parameters



### 5.1.2 Inter Packet Delay

The *Inter Packet Delay* is usually applied when multiple cameras are connected to one PC over the same Network Interface Card (NIC). It enables the user to create a pause between consecutive packets which reduces the amount of effective load and creates timeslots for packets from other devices on the connection.

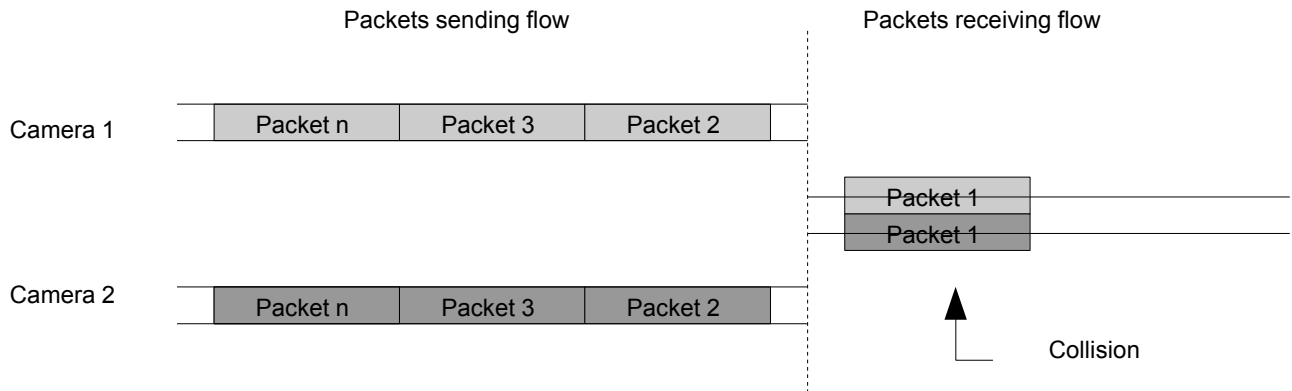


Figure 111: Packet flow while not using inter packet delay

If the *Inter Packed Delay* is not used, excessive collision between packets may occur which results in data loss, like illustrated in Figure 111. Packets from two cameras are sent to PC over the same network connection. Without any *Inter Packet Delay* set, collision between packets from different cameras may occur in case of insufficient bandwidth. Figure 112 illustrates a well configured *Inter Packet Delay* to prevent collisions.

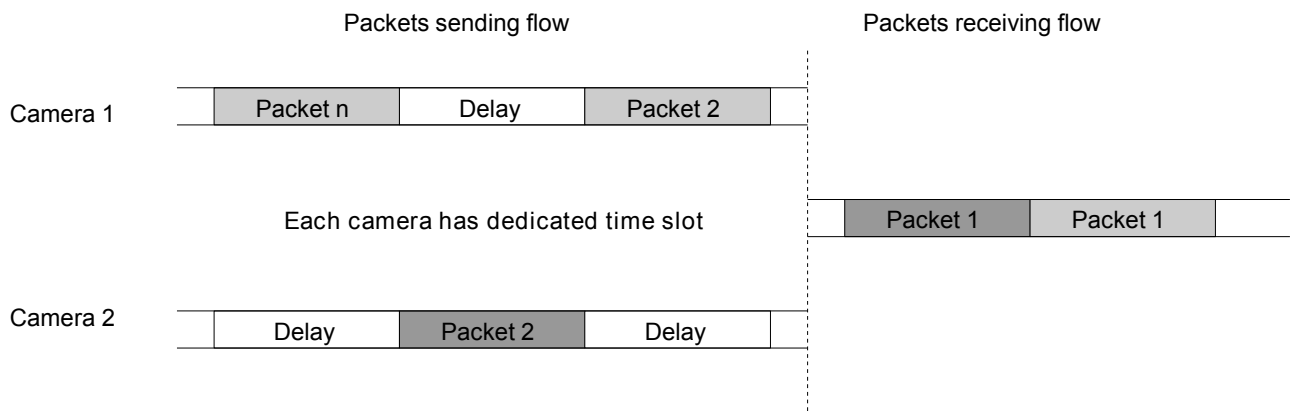


Figure 112: Packet flow while using inter packet delay

### Setting Inter Packet Delay

The cameras provide the features *GevSCPSPacketSize* and *GevSCPD*. *GevSCPD* represents the Inter Packet Delay and is expressed in microseconds. Its value can range from 0 to 1000 $\mu$ s and should be set according to number of cameras connected to a certain network interface card and *GevSCPSPacketSize*.

The *GevSCPSPacketSize* feature represents the size of packets and is expressed in bytes, the default camera packet size is at 1500 bytes, but can be larger if the network hardware supports Jumbo Frames.

Assuming that the *GevSCPSPacketSize* is 1500 bytes (effective Ethernet packet size including inter-frame gap, preamble, header and CRC on the wire is 1538 bytes), maximum of 81274 packets are sent every second via the Ethernet interface. It takes 8ns to transfer one byte over Gigabit Ethernet network, so time required to transfer one packet of 1538 bytes is 12,3 $\mu$ s. The *GevSCPD* should be a bit longer than the time required to transfer one packet, in order to ensure that packets from second camera will fit in the vacant time slot. On the other hand, if the camera is producing 60000 packets per second (50 frames per second, 1200 packets per frame), total transfer time must not exceed 16,67 $\mu$ s if frame rate is to be preserved.

### Example

Three cameras are connected to one PC, and are sending 1500 byte packets each. *GevSCPD* should be such that packets from all three cameras are serialized to the PC's Network Interface Card. Setting inter packet delay to 25 $\mu$ s (12,3 $\mu$ s + 12,3 $\mu$ s  $\approx$  25 $\mu$ s) will ensure that packets from other two cameras will fit in the gap between two consecutive packets.

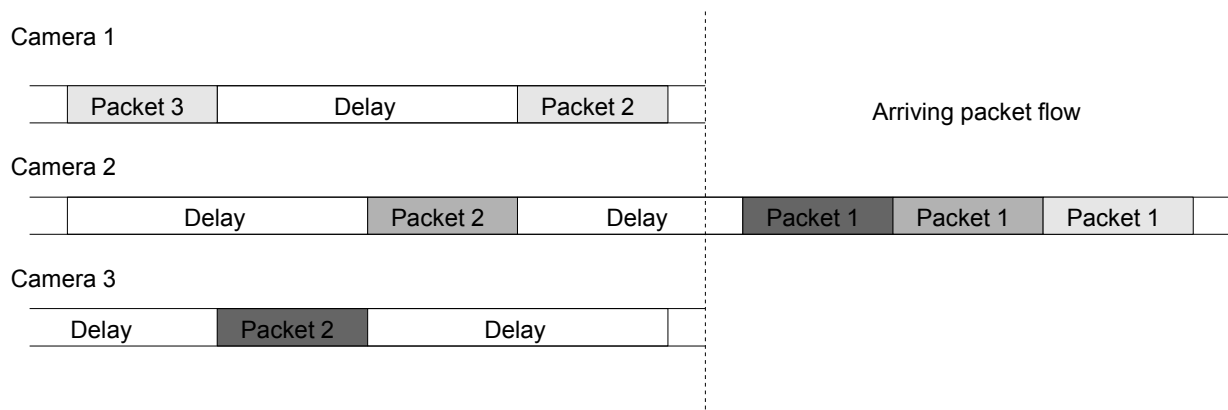


Figure 113: Packet flow example with three cameras and inter packet delay

### 5.1.3 Frame Transfer Delay

The *Frame Transfer Delay* sets the frame transfer start delay (in ticks) for the selected stream channel. This value represents a delay between the point in time when a frame is ready for transmission and when transmission actually starts. *Frame Transfer Delay* feature is useful in situations where there are many simultaneously triggered cameras on the network requiring more bandwidth than is available. In such scenario network can become overwhelmed with incoming data and start losing packets triggering packet resend mechanism.

To calculate required *Frame Transfer Delay* use the formula shown below:

$$\text{FrameTransferDelay} = \text{numberOfPackets} \times \text{packetTransferTime}$$

*FrameTransferDelay* - Frame Transfer Delay expressed in time unit [ns]

*packetTransferTime* - time required to transfer a packet over the network

*numberOfPackets* - amount of packets contained in one frame

$$\text{packetTransferTime} = \text{byteTransferTime} * \text{totalBytesInPacket}$$

*byteTransferTime* - time to transfer one byte over the network. It is 8ns on Gigabit Ethernet network

*totalBytesInPacket* - total number of bytes transferred in one packet

$$\text{totalBytesInPacket} = \text{GevSCPSPacketSize} + \text{sizeofEthernetHeaders}$$

where *sizeofEthernetHeaders* is 38 bytes which includes inter-frame gap, preamble, header and CRC.

$$\text{numberOfPackets} = \frac{\text{PayloadSize}}{\text{effectiveBytesInPacket}}$$

*Payloadsize* - frame size in bytes (retrieved from camera)

*effectiveBytesInPacket* - number of effective bytes transferred in packet without headers

$$\text{effectiveBytesInPacket} = \text{GevSCPSPacketSize} - (\text{IPHeaderSize} + \text{UDPHeaderSize} + \text{GVSPHeaderSize})$$

where *IPHeaderSize* is 20 bytes, *UDPHeaderSize* is 8 bytes and *GVSPHeaderSize* is 8 bytes.

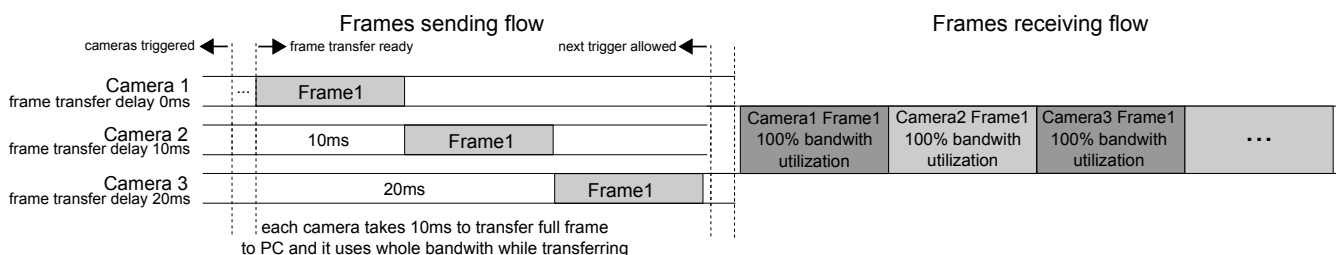


Figure 114: Example flow of frames when GevSCFTD is used

Figure 114 shows a case where three simultaneously triggered cameras are streaming frames to one PC and each camera utilizes 100% of available bandwidth when transferring frame. In this particular sample it takes 10ms to transfer one whole frame for each camera so *Frame Transfer Delay* needs to be adjusted in a way that only one camera is transferring data at a time. In presented case Camera 2 will start sending data after frame from Camera 1 is transferred and Camera 3 will start sending data after frame from Camera 2 is transferred. Next trigger is not allowed until all cameras finish sending data.

### Setting Frame Transfer Delay

For setting *FrameTransferDelay* property, camera provides *GevSCFTD* register which is represented in ticks. To calculate the value for *GevSCFTD*, *FrameTransferDelay* needs to be converted from time unit to ticks. Before calculating, *FrameTransferDelay* need to be converted to seconds so correct value can be calculated. Formula to calculate number of ticks for given time value is shown below:

$$\text{GevSCFTD} = \text{FrameTransferDelay} \times \text{GevTimestampTickFrequency}$$

*GevTimestampTickFrequency* - indicates the number of timestamp ticks during 1 second

## 5.2 LAN IP Configuration

To successfully establish the connection to a camera, the Network configuration needs to be done according to the requirements of the application. The connection of the physical Network Interface Card (NIC) needs to be enabled and set-up properly. Table 73 shows a good configuration for a first start with a single NIC, but does not represent a set definition for all application and network environments.

	NIC (Cameras)	NIC (others)	Camera 1	Camera 2	Camera 3
<b>IP</b>	169.254.0.1	not 169.254.x.x	169.254.1.1	169.254.1.2	169.254.1.x
<b>Subnetmask</b>	255.255.0.0	if 255.255.0.0	255.255.0.0	255.255.0.0	255.255.0.0

Table 73: Basic IP configuration of the PC




### Note

To using several cameras on multiple Network Interface Cards (NIC) in one PC, make absolutely sure that each NIC is configured for a different network. Otherwise it will not be possible to operate all cameras correctly.

## IP Setup in Microsoft Windows

On Microsoft Windows operating systems, the IP can be set up by the following steps:

1. Execute "ncpa.cpl", e.g. via the command box of the Windows *Startmenu* or after pressing  + R on your keyboard
2. Right-click the target network interface card and choose *Properties*.
3. Choose the *Internet Protocol Version 4 (TCP/IPv4)* entry in the list of protocols.
4. By default the interface is configured to obtain an IP address automatically by a DHCP server, shown in Figure 115, usually provided by routers or dedicated servers. A fixed IP address and Subnet mask can be entered like shown in Figure 115 right, after choosing *Use the following IP address*.

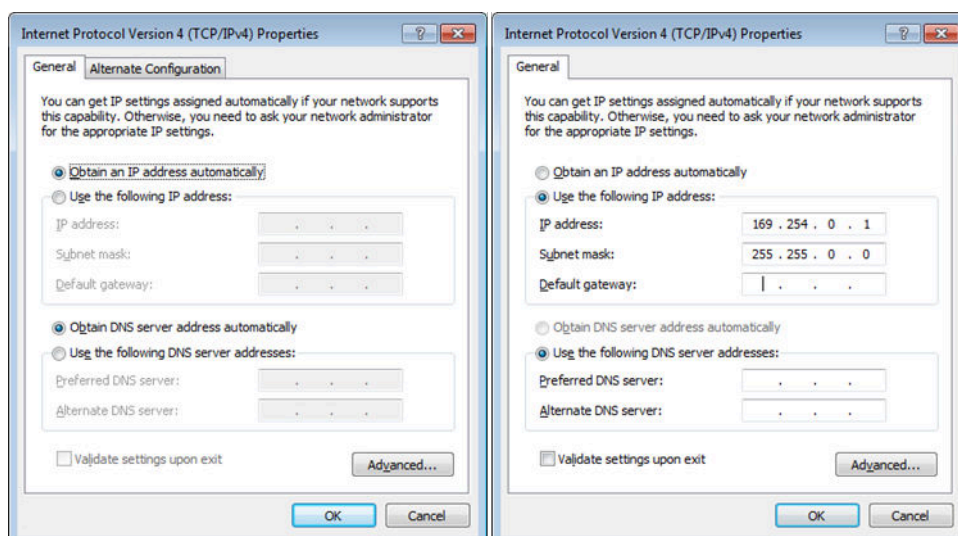


Figure 115: Internet Protocol Version 4 (TCP/IPv4) properties

### 5.3 Network Interface Optimization

To reach the optimal performance with the Giganetix cameras, the right choice of hardware is crucial, as well as its configuration. All following descriptions of driver settings are based on the Intel® network driver interface on Microsoft Windows, the availability of each feature and its limits can differ between hardware vendors and operating systems.

#### 5.3.1 Choosing the right Network Interface Card

The first step is to choose the right network interface card (NIC). It is strongly recommended to use *PCI Express* based Gigabit Ethernet NICs supporting so called *Jumbo Frames* or *Jumbo Packages*. NICs based on the old interface standard *PCI* do often not provide enough guaranteed bandwidth due to the limited and shared bandwidth of the *PCI* bus. However, the *PCI Express* bus provides guaranteed enough bandwidth for Gigabit Ethernet. Jumbo frames reduce the overhead and workload on the target PC, reducing the amount of packets to be processed by sending a smaller count of larger packets instead of a high count of small packets. A good choice are NICs with the Intel® Pro/1000 chipset, like the Intel® Pro/1000 CT Desktop (single port) or PT Server Adapter (multiport).



#### Note

Jumbo packets / frames need to be supported by the NIC as well as the whole network chain, including all network switches passed on the route.


#### 5.3.2 Using Jumbo Frames / Packets

As soon as Jumbo Frames / Packets are supported by the network infrastructure, it has to be made sure that this feature is enabled on each involved device. Devices like NICs have this feature usually disabled by default as well as some managed switches, what makes it suggestive to throw a look into their configuration.

#### Network Interface Card (NIC)

For NICs it is usually necessary to enable the Jumbo Frame / Packet feature manually within the device driver.

On Microsoft Windows operating systems this can be accessed on the following way:

1. Execute "ncpa.cpl", e.g. via the command box of the Windows *Startmenu* or after pressing  + R on your keyboard
2. Right-click the target network interface card and choose *Properties*.
3. Press the *Configure* button below the name of the physical network interface, shown in Figure 116.

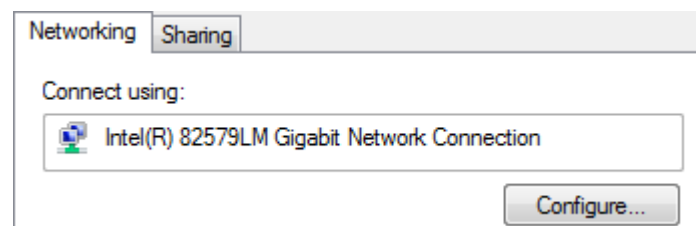


Figure 116: Access to network interface card driver settings

4. The settings of the driver can be accessed by choosing the *Advanced* tab of the opened window. As shown in Figure 117, raise the value of the *Jumbo Packet* property to its maximum value.

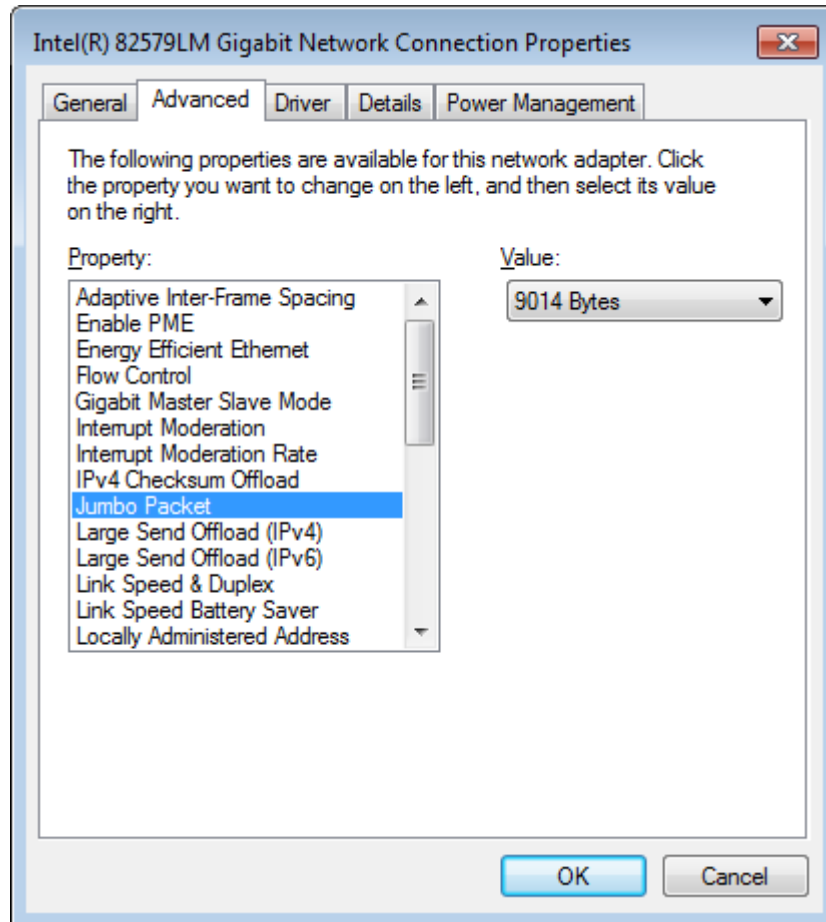


Figure 117: Network interface card - advanced driver settings - Jumbo Packets


### Gigabit Ethernet Switches

Standard or simple network switches (unmanaged) which support Jumbo Frames / Packets usually have this feature enabled per default, as they offer no way of configuration. Professional or so called managed switches, which provide a configuration interface (in most cases with a web based GUI), have it in many cases disabled as it is configurable for each port separately. For validation please refer to the documentation of your individual device.

### 5.3.3 Raising Receive Buffers

The receive buffer size of the network interface card represents a reserved memory in the system memory, which is used to buffer incoming data. Especially at the high bandwidth of Gigabit Ethernet cameras, it is recommended to raise the buffer size for at least the incoming packets (receive buffers) to the supported maximum. As it is a feature of the network interface card, it usually can be accessed via its driver settings.

On Microsoft Windows operating systems it can be accessed on the following way:

1. Execute "ncpa.cpl", e.g. via the command box of the Windows *Startmenu* or after pressing  + R on your keyboard
2. Right-click the target network interface card and choose *Properties*.
3. Press the *Configure* button below the name of the physical network interface, shown in Figure 116.
4. The settings of the driver can be accessed by choosing the *Advanced* tab of the opened window. As shown in Figure 118, raise the value of the *Receive Buffers* property to its maximum value.

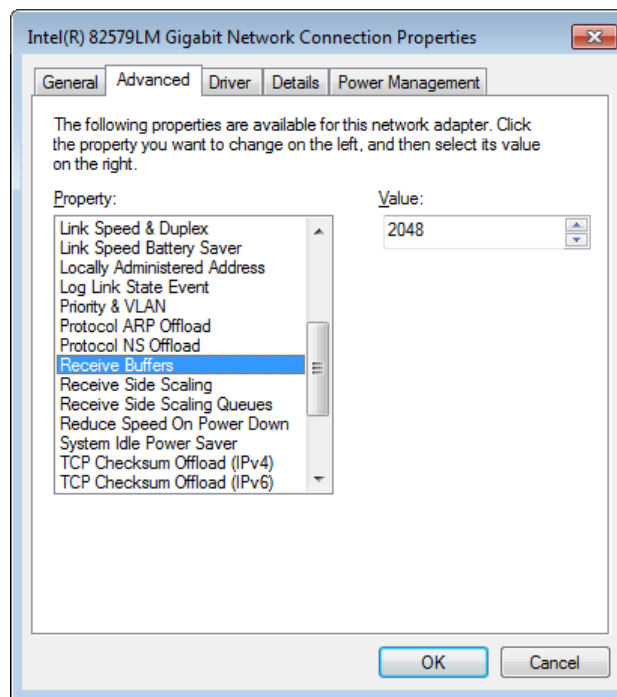



Figure 118: Network interface card - advanced driver settings - Receive Buffers



### 5.3.4 Disable the Interrupt Moderation Rate

To optimize the latency of the data transmission, it is recommended to disable the *Interrupt Moderation Rate* feature of the network adapter. This feature changes the way of interrupt generation by an incoming packet and makes it possible to reduce the delay in the processing of incoming Ethernet packets. It is usually accessible via the identically named property in the advanced driver settings of the NIC, shown in Figure 119.

On Microsoft Windows operating systems it can be accessed on the following way:

1. Execute "ncpa.cpl", e.g. via the command box of the Windows *Startmenu* or after pressing  + R on your keyboard
2. Right-click the target network interface card and choose *Properties*.
3. Press the *Configure* button below the name of the physical network interface, shown in Figure 116.
4. The settings of the driver can be accessed by choosing the *Advanced* tab of the opened window. As shown in Figure 119, set the value of the *Interrupt Moderation Rate* property to *Off* or *Disabled*.

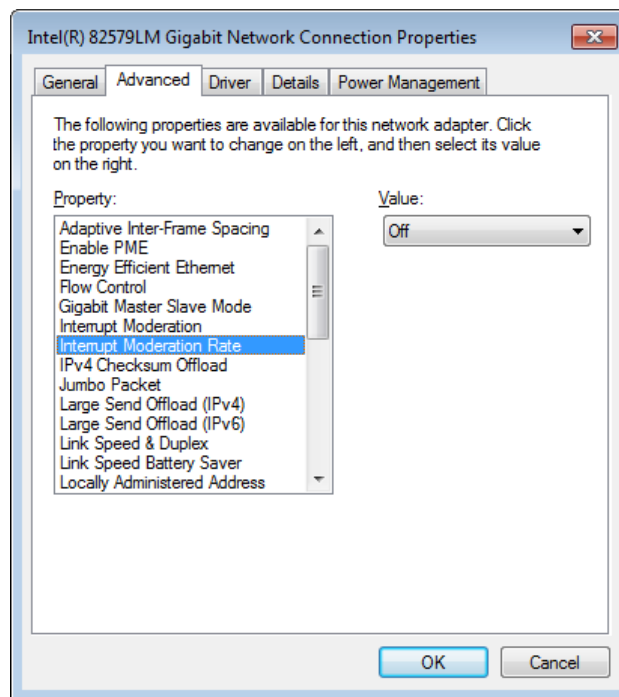


Figure 119: Network interface card - advanced driver settings - Interrupt Moderation Rate

### 5.3.5 Disable the Flow Control


To further optimize the latency of the data transmission, shown in 5.3.4 - *Disable the Interrupt Moderation Rate*, it is possible to disable the so called *Flow Control*. The *Flow Control* is a feature to adapt the transmission speed of the transmitter to the speed available at the receiver by sending PAUSE frames in-between.



#### Note

As deactivating this feature can cause in packet lost due to missing bandwidth, it is generally not recommended to be disabled!

On Microsoft Windows operating systems the Flow Control can be accessed on the following way:

1. Execute "ncpa.cpl", e.g. via the command box of the Windows *Startmenu* or after pressing  + R on your keyboard
2. Right-click the target network interface card and choose *Properties*.
3. Press the *Configure* button below the name of the physical network interface, shown in Figure 116.
4. The settings of the driver can be accessed by choosing the *Advanced* tab of the opened window. As shown in Figure 120 , set the value of the *Interrupt Moderation Rate* property to *Disabled*.

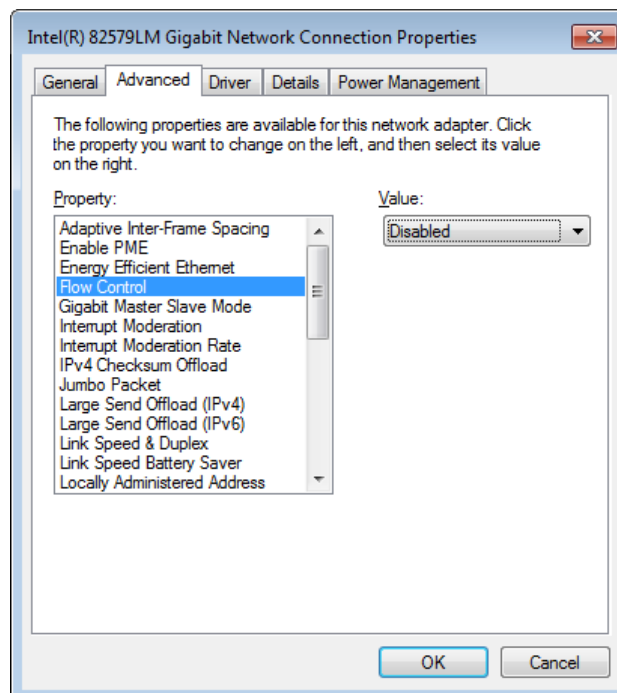


Figure 120: Network interface card - advanced driver settings - Flow Control

## 5.4 Digital Image and Pixel Formats

This section describes pixel and image layouts supported by SMARTEK Vision digital cameras. While a pixel format describes how a single pixel data is constructed, the image layout represents how the data are ordered in the memory of the captured device. It is identical for all cameras.

### 5.4.1 Image Layout

Figure 121 shows the image layout valid for all SMARTEK Vision digital cameras. An image transmitted out of a camera is considered to be a two-dimensional array of pixels. The pixels are stored in subsequent addresses in the memory of the capture device, usually a PC. Each pixel, depending on its format, is either 8 bits or 16 bits wide. It is described by two indices; the first index indicates the row while the second index indicates the column where the pixel is located:

- $P(x, y)$  means the pixel located at row  $x$  and at column  $y$ .
- $P(1, 1)$  means the pixel located at row 1 and at column 1.
- $P(1, 2)$  means the pixel located at row 1 and at column 2.

An image with a width of  $w$  and a height of  $h$  starts at the upper left corner and ends at the bottom right corner.  $P(1, 1)$  is the first pixel of the image,  $P(h, w)$  is the last pixel of the image.

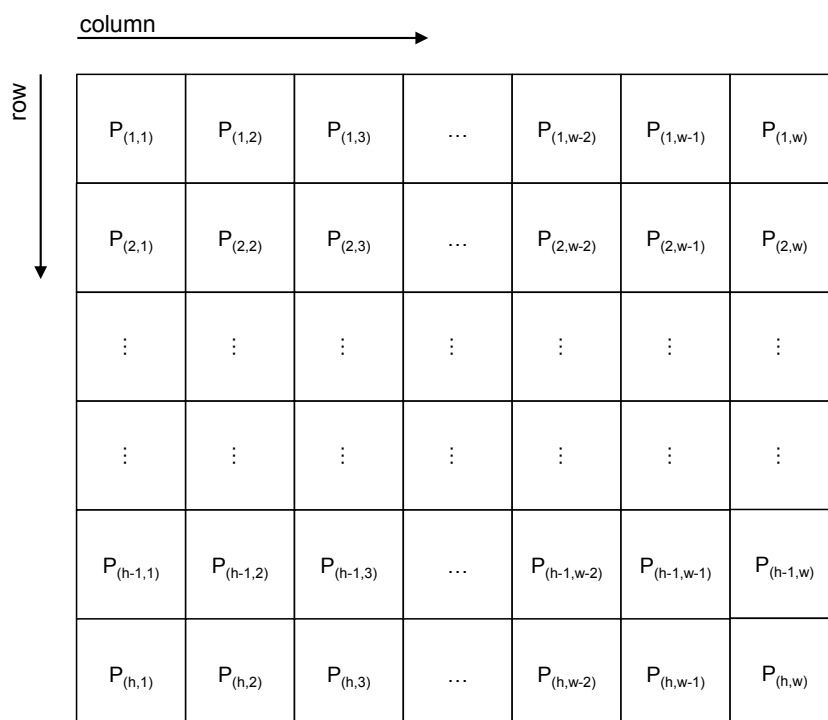


Figure 121: Image layout and transmission sequence

The subsequent sections will describe all the supported formats for each pixel  $P$  in an image, for monochrome cameras as well as color cameras.

## 5.4.2 Supported Pixel Formats for Monochrome Cameras

### 5.4.2.1 Mono8

In an image with the pixel format Mono8 each pixel value  $P$  is represented by one byte or 8 bits. The Mono8 pixel format in SMARTEK Vision digital cameras is specified as shown below:

PixelFormat	Mono8
Description	8-bit monochrome unsigned
Pixel size	1 byte
Value range	0 ... 255

Table 74: Specification PixelFormat Mono8

The memory layout of the image with *Mono8* pixel format is shown in Figure 122. Starting from the upper left of the image, byte 0 represents the value of pixel  $P(1, 1)$ , byte 1 represents the value of pixel  $P(1, 2)$  and so on. In each byte the bitorder is by default little endian; the least significant bit is assigned to bit 0 and the most significant bit to bit 7.

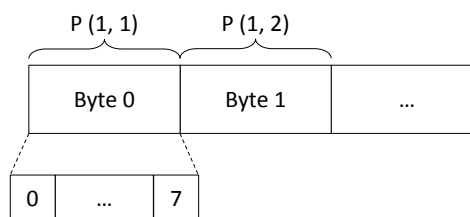


Figure 122: Image layout with pixel format Mono8

### 5.4.2.2 Mono10Packed

In an image with the pixel format Mono10Packed each two pixel values are represented by three bytes or 24 bits. The Mono10Packed pixel format in SMARTEK Vision digital cameras is specified as shown below:

PixelFormat	Mono10Packed
Description	10-bit packed monochrome unsigned
Pixel size	3 bytes for two pixels
Value range	0 ... 1023

Table 75: Specification PixelFormat Mono10Packed

The memory layout of the image with *Mono10Packed* pixel format is shown in Figure 123. Starting from the upper left of the image, byte 0 and first 2 bits of byte 1 represents the value of pixel  $P(1, 1)$ . Bits 5 and 6 in byte 1 together with all bits from byte 2 represents the value of pixel  $P(1, 2)$  and so on. In each byte the bitorder is by default little endian; the least significant bit is assigned to bit 0 and the most significant bit to bit 9.

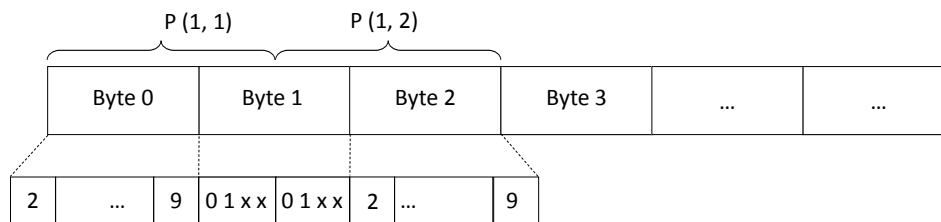


Figure 123: Image layout with pixel format Mono10Packed

### 5.4.2.3 Mono12Packed

In an image with the pixel format Mono12Packed each two pixel values  $P$  are represented by three bytes or 24 bits. The Mono12Packed pixel format in SMARTEK Vision digital cameras is specified as shown below:

PixelFormat	Mono12Packed
Description	12-bit packed monochrome unsigned
Pixel size	3 bytes for two pixels
Value range	0 ... 4095

Table 76: Specification PixelFormat Mono12Packed

The memory layout of the image with *Mono12Packed* pixel format is shown in Figure 124. Starting from the upper left of the image, byte 0 and first 4 bits of byte 1 represents the value of pixel  $P(1, 1)$ . Second half of bits in byte 1 together with all bits from byte 2 represents the value of pixel  $P(1, 2)$  and so on. In each byte the bitorder is by default little endian; the least significant bit is assigned to bit 0 and the most significant bit to bit 11.

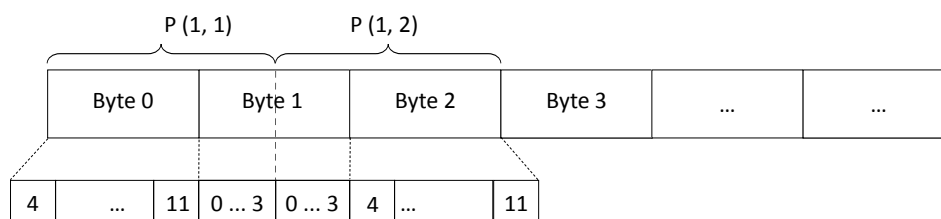


Figure 124: Image layout with pixel format Mono12Packed

### 5.4.2.4 Mono16

In an image with pixel format *Mono16* each pixel value  $P$  is represented by two bytes or 16 bits. The *Mono16* pixel format in SMARTEK Vision digital cameras is specified as shown below:

PixelFormat	Mono16
Description	16-bit monochrome unsigned
Pixel size	2 byte
Value range	0 ... 65535

Table 77: Specification PixelFormat Mono16

The two bytes are arranged in little-endian order, which means that the Least Significant Byte (LSB) is arranged first, the most significant byte second. The memory layout of the image with the Mono16 pixel format is shown in Figure 125. Starting with the upper left of the image, byte 0 and byte 1 represent the value of pixel P(1, 1), byte 2 and byte 3 represent the value of pixel P(1, 2) and so on. The least significant bit is assigned to bit 0 of the first byte, the most significant bit to bit 7 of the second byte.

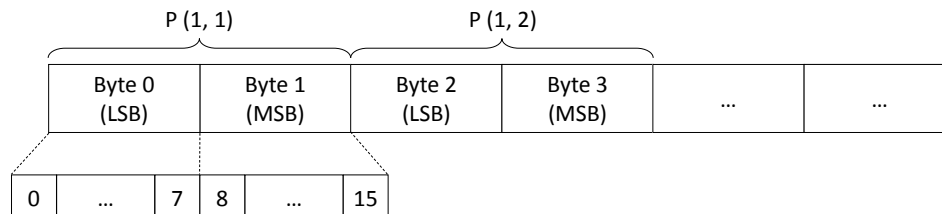


Figure 125: Image layout with pixel format Mono16

### 5.4.3 Supported Pixel Formats for Color Cameras

#### 5.4.3.1 BayerGR8 / BayerRG8 / BayerGB8 / BayerBG8

In an image with one of the *Bayer8* pixel formats, each pixel value P is represented by one byte or 8 bits. The "GR", "RG", "GB" or "BG" notation describes the layout of the Bayer pattern on the image sensor, used in the camera. For detailed description about color imaging and the Bayer filter, please refer to chapter 4.1.5 - *Color Imaging with Bayer Pattern*.

The *Bayer8* pixel formats in SMARTEK Vision digital cameras are specified like shown below:

PixelFormat	BayerGR8, BayerRG8, BayerGB8, BayerBG8
Description	8-bit monochrome unsigned
Pixel size	1 byte
Value range	0 ... 255

Table 78: Specification PixelFormat Bayer8

The memory layout of the image with this pixel formats is shown in Figure 126. Starting from the upper left of the image, byte 0 represents the value of pixel P(1, 1), byte 1 represents the value of pixel P(1, 2) and so on. In each byte the bitorder is by default little endian; the least significant bit is assigned to bit number 0 and the most significant bit is assigned to bit number 7.

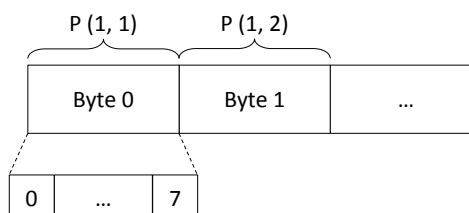


Figure 126: Image Layout with pixel format GR8/RG8/GB8/BG8

### 5.4.3.2 BayerGR16 / BayerRG16 / BayerGB16 / BayerBG16

In an image with pixel format *Bayer16* each pixel value *P* is represented by two byte or 16 bits. The "GR", "RG", "GB" or "BG" notation describes the Bayer pattern of the image sensor used in the camera. For detailed description about the Bayer filter, please refer to chapter 4.1.5 - Color Imaging with Bayer Pattern. The *Bayer16* pixel format in SMARTEK Vision digital cameras is specified like shown below:

PixelFormat	BayerGR16, BayerRG16, BayerGB16, BayerBG16
Description	16-bit monochrome unsigned
Pixel size	2 byte
Value range	0 ... 65535

Table 79: Specification PixelFormat Bayer16

The two bytes are arranged in little-endian order, which means the Least Significant Byte comes first, the most significant byte comes second. The memory layout of the image with the *Bayer16* pixel format is shown in Figure 127. Starting from the upper left of the image, byte 0 and byte 1 represent the value of pixel *P*(1, 1), byte 2 and byte 3 represent the value of pixel *P*(1, 2) and so on. The least significant bit is assigned to bit 0 of the first byte, the most significant bit to bit 7 of the second byte.

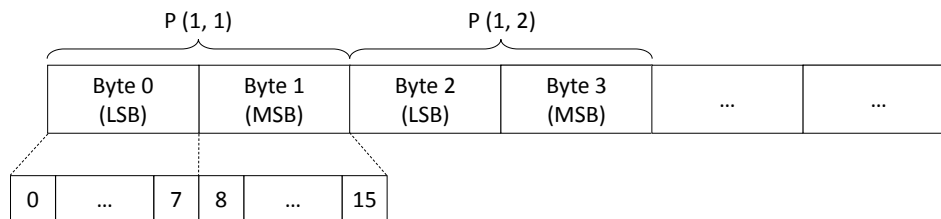


Figure 127: Image layout with pixel format GR16/RG16/GB16/BG16

#### 5.4.4 Pixel Formats Supported by the SMARTEK Vision Giganetix camera family

Monochrome Cameras				
CMOS	Mono8	Mono10Pack	Mono12Pack	Mono16
GC1281M	●	○	○	●
GC2591M	●	○	○	●
GC3851M	●	○	○	●
GC1932M	●	●	○	●
GCP1931M	●	●	●	●
GCP2061M	●	●	●	●
GCP2461M	●	●	●	●
CCD	Mono8	Mono10Pack	Mono12Pack	Mono16
GC651M	●	○	○	●
GC652M	●	○	○	●
GC653M	●	○	○	●
GC781M	●	○	○	●
GC1031M	●	○	○	●
GC1291M	●	○	○	●
GC1391M	●	○	○	●
GC1392M	●	○	○	●
GC1621M	●	○	○	●
GC2441M	●	○	○	●
GC1021M	●	○	○	●
GC1601M	●	○	○	●
GC1921M	●	○	○	●
GCP1941M	●	●	●	●
GCP2751M	●	●	●	●
GCP3381M	●	●	●	●
GCP4241M	●	●	●	●

Table 80: Pixel formats supported by SMARTEK Vision monochrome cameras



Color Cameras				
CMOS	BayerGR8	BayerGR16	BayerRG8	BayerRG16
GC2041C	●	○	○	○
GC2591C	●	○	○	○
GC3851C	●	○	○	○
GC1932C	○	○	●	●
GCP1931C	○	○	●	●
GCP2061C	○	○	●	●
GCP2461C	○	○	●	●
CCD	BayerGR8	BayerGR16	BayerRG8	BayerRG16
GC651C	○	○	●	●
GC652C	○	○	●	●
GC653C	○	○	●	●
GC781C	○	○	●	●
GC1031C	○	○	●	●
GC1291C	○	○	●	●
GC1391C	○	○	●	●
GC1392C	○	○	●	●
GC1621C	○	○	●	●
GC2441C	○	○	●	●
GC1021C	●	●	○	○
GC1601C	●	●	○	○
GC1921C	●	●	○	○
GCP1941C	○	○	●	●
GCP2751C	●	●	○	○
GCP3381C	○	○	●	●
GCP4241C	○	○	●	●

Table 81: Pixel formats supported by SMARTEK Vision color cameras



#### Note

On dual tap CCD camera models, 16 bit pixel formats are supported by the means of firmware update. There are two different firmware versions for those camera models: one supporting 8 bit pixel formats and the other supporting 16 bit pixel formats.



### Note

The 16 Bit formats of SMARTEK Vision Giganetix cameras contain the full, in chapter 2.2 - Sensor Information and Technical Specification (All Models Separate) specified, bit depth available by the sensor. As this are in most cases less than 16 Bit of payload, the effective image data is aligned to the Most Significant Bits (MSB).

## 5.5 Chunk Data

*Chunk Data* are information related to the image, captured and transmit by a GigE Vision camera. As soon as active, the camera adds different kinds of information to each captured image. For instance this information can be the image itself as well as data which has been extracted from the image, or configuration parameters like the area of interest configuration, pixel format, state of I/O pins and user defined data.

Each chunk can be individually enabled or disabled. For example if frame counter chunk is enabled, after each captured image, the camera checks the frame counter register and sends its value together with the image as frame counter chunk. Chunks are, together with captured image, fitted into tagged blocks of data.

Each data block consists of a *Data Leader*, *Payload* and *Trailer*. The *Data Leader* is one packet that determines the beginning of a new data block. The *Data Payload* are one or more packets containing the actual information to be streamed for the current data block. The *Data Trailer* is one packet used to signal the end of the data block. Chunk data is also included in the *Data Payload*.

Each chunk contains chunk data and the trailing tag. The trailing tag contains a unique chunk identifier and the length of the chunk data. The unique chunk identifier defines the structure of the chunk data and the chunk feature associated with this chunk. Figure 128 shows the chunk data content.

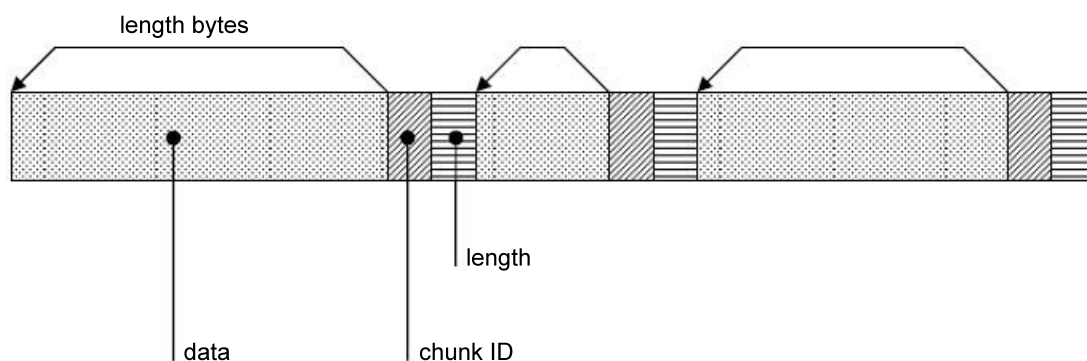


Figure 128: Chunk data content

- **data:** The data that the chunk is transporting. This section must be a multiple of 4 bytes; empty digits have to be padded with zeros. This ensures the chunk ID and length fields are 32-bit aligned with chunk.
- **chunk ID:** The chunk identifier (4 bytes)
- **length:** The length of the data in bytes, must be a multiple of 4

### 5.5.1 Getting Started with Chunk Data

Before using any of the chunk data features, chunk mode should be checked to be supported by the camera. The following C++ code shows how to check if *ChunkMode* is available:

```
device->IsAvailable("ChunkModeActive")
```

If chunk data is available, it can be enabled by the *ChunkMode* property:

```
device->SetBooleanNodeValue("ChunkModeActive", true)
```

*ChunkMode* can be disabled by setting the *ChunkModeActive* feature to *false*. By enabling *ChunkMode*, *ImageInfo* chunk is automatically enabled and included into the image payload data (this chunk cannot be disabled as long as *ChunkMode* is active). Also all additional chunks are made available for inclusion.

The appended *ImageInfo* chunk contains basic informations about the image and is required to correctly display acquired images. The attributes included in the *ImageInfo* chunk are shown in Table 82.

Chunk Feature	Description
<i>ChunkPixelFormat</i>	Pixel format of the current image
<i>ChunkLinePitch</i>	Line size in bytes
<i>ChunkWidth</i>	Width of the image
<i>ChunkHeight</i>	Height of the image
<i>ChunkOffsetX</i>	Offset x of the image
<i>ChunkOffsetY</i>	Offset y of the image

Table 82: Basic chunks

Further, additional chunks can optionally be included into the image by following the steps below:  
First desired chunk needs to be selected:

```
device->Set(TYPE)NodeValue("ChunkSelector", "ChunkName")
```

Afterward it needs to be enabled:

```
device->SetBooleanNodeValue("ChunkEnable", true)
```

A list of chunk features is shown in Table 83.

Chunk Feature	Description
<i>ChunkGain</i>	Gain applied to the image
<i>ChunkExposureTime</i>	Exposure time of the image
<i>ChunkCounterValue</i>	Selected counter value
<i>ChunkUserIntValue</i>	User defined integer value

Table 83: Additional chunks



**Note** The *PayloadType* of the camera changes from *PayloadImage* to *PayloadChunkData*.

On receiving images containing chunk data, the chunk information can be read using the following code:

```
if (imageInfo->IsReadable("ChunkNAME"))
TYPE value;
imageInfo->GetChunkNode("ChunkNAME")->GetTYPENodeValue(value);
```

By reading chunks this way, it can be made sure that it is included into the image before it is tried to be accessed and what makes sure that application will not crash if chunk is accidentally not sent.

Table 84 shows important C++ functions related to chunk data, a full description of interface and further supported languages can be found in the API documentation located in the GigEVisionSDK installation folder.

Function	Description
<code>device-&gt;IsAvailable("ChunkModeActive")</code>	Returns true if ChunkMode is available or false if not.
<code>device-&gt;SetBooleanNodeValue("ChunkModeActive", value)</code>	Enable or disable ChunkMode.
<code>device-&gt;Set(TYPE)NodeValue("ChunkSelector", "ChunkName")</code>	Select chunk to enable or disable.
<code>device-&gt;SetBooleanNodeValue("ChunkEnable", double value)</code>	Enable or disable selected chunk.
<code>imageInfo-&gt;IsReadable("ChunkName")</code>	Returns true if user can access selected chunk and read data from it.
<code>imageInfo-&gt;GetChunkNode-&gt;("ChunkNAME")-&gt;GetTYPENodeValue(value)</code>	Used to read data from chunk. User should first check if he can access selected chunk.

Table 84: Chunk mode - Access through API

Table 85 contains functions showing how to retrieve data from each chunk.

Chunk name	Type	Function
ChunkPixelFormat	enumeration	<code>imageInfo-&gt;GetChunkNode-&gt; ("ChunkPixelFormat")-&gt;GetEnumNodeValue (value)</code>
ChunkLinePitch	integer	<code>imageInfo-&gt;GetChunkNode-&gt; ("ChunkLinePitch")-&gt;GetIntegerNodeValue (value)</code>
ChunkWidth	integer	<code>imageInfo-&gt;GetChunkNode-&gt; ("ChunkWidth")-&gt;GetIntegerNodeValue (value)</code>
ChunkWidth	integer	<code>imageInfo-&gt;GetChunkNode-&gt; ("ChunkWidth")-&gt;GetIntegerNodeValue (value)</code>
ChunkHeight	integer	<code>imageInfo-&gt;GetChunkNode-&gt; ("ChunkHeight")-&gt;GetIntegerNodeValue (value)</code>
ChunkOffsetX	integer	<code>imageInfo-&gt;GetChunkNode-&gt; ("ChunkOffsetX")-&gt;GetIntegerNodeValue (value)</code>
ChunkOffsetY	integer	<code>imageInfo-&gt;GetChunkNode-&gt; ("ChunkOffsetY")-&gt;GetIntegerNodeValue (value)</code>
ChunkGain	float	<code>imageInfo-&gt;GetChunkNode-&gt; ("ChunkGain")-&gt;GetFloatNodeValue (value)</code>
ChunkExposureTime	float	<code>imageInfo-&gt;GetChunkNode-&gt; ("ChunkExposureTime")-&gt;GetFloatNodeValue (value)</code>
ChunkCounterValue	integer	<code>imageInfo-&gt;GetChunkNode-&gt; ("ChunkCounterValue")-&gt;GetIntegerNodeValue (value)</code>
ChunkUserIntValue	integer	<code>imageInfo-&gt;GetChunkNode-&gt; ("ChunkUserIntValue")-&gt;GetIntegerNodeValue (value)</code>

Table 85: Access to chunks through API

### 5.5.2 Additional chunks

#### 5.5.2.1 ChunkGain

Gain chunk contains the *Gain* value of the image that is included in same payload.  
To enable *Gain* chunk:

- use *ChunkSelector* to select *GainChunk*
- set *ChunkEnable* to *true*

#### 5.5.2.2 ChunkExposureTime

ExposureTime chunk contains *ExposureTime* of the image that is included in the same payload.  
To enable *ExposureTime* chunk:

- use *ChunkSelector* to select *ExposureTimeChunk*
- set *ChunkEnable* to *true*

#### 5.5.2.3 ChunkCounterValue

Counter chunk can display values from different counter sources, as e.g. the *FrameCounter*. *FrameCounter* is a 32 bit value which starts at 0 and increments by 1 for each acquired frame (image). The maximum value of *FrameCounter* is 4294967295, after which it will reset to 0 again and continue counting. *FrameCounter* can be set to any positive value from which it will continue counting.

The following code shows how to set *CounterValue*:

```
device->SetIntegerNodeValue("CounterValue", value)
```

To enable *CounterChunk*:

- use *ChunkSelector* to select the appropriate counter chunk
- set *ChunkEnable* to *true*

### 5.5.2.4 ChunkUserIntValue

The *ChunkUserIntValue* is an integer value defined by the user. When *ChunkUserIntValue* is enabled, a chunk is added to each frame containing the value of the *UserIntValue* property. The *ChunkUserIntValue* is a 4 byte value.

The following code shows how to set *UserIntValue*:

```
device->SetIntegerNodeValue("UserIntValue", value)
```

To enable *UserIntValue* chunk:

- use *ChunkSelector* to select *UserIntValue* chunk
- set *ChunkEnable* to *true*



## 6 Image Processing on Camera

This section will describe image processing algorithms supported on SMARTEK Vision cameras. Currently image processing on camera is supported only on specific camera models.

### 6.1 Luminance Look-up Table

The GCP series equipped with monochrome sensors support 12 bit look-up table. The theory and applications of look-up table will be introduced in chapter 7.2.1.

Figure 129 shows the on camera look-up table feature located in the *LUT-CAM* tab within the *Image Processing Properties* panel. If not visible, the *Image Processing Properties* panel can be activated by the menu bar entry *Control*  $\Rightarrow$  *Image Processing Properties*.

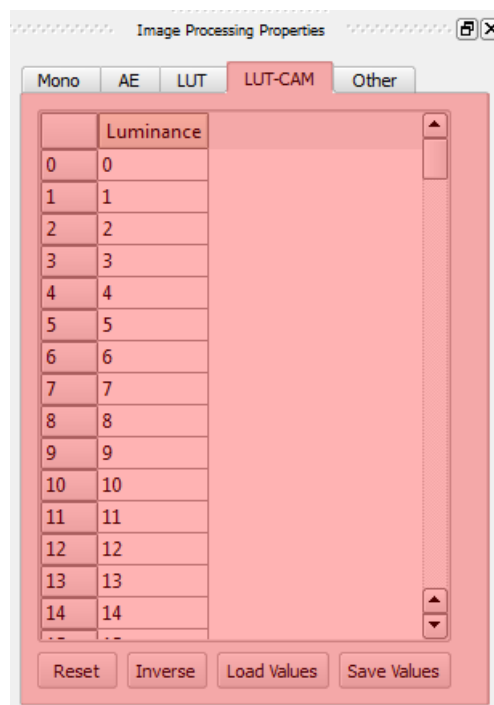


Figure 129: LUT-CAM feature in GigEVisionClient

- **Reset:** Reset look-up table to default values
- **Inverse:** Generate a predefined look-up table which inverts the image
- **Load Values:** Load an user-defined XML file with look-up table parameters into the client
- **Save Values:** Save the user-defined look-up table to a file

A common way to set all LUT values at a time in the client is to describe the LUT in a XML file and load it to the camera. To accomplish this, use the *Load Values* feature in the *GigEVisionClient* (see Figure 129). Please refer to the Look-up table section in chapter 7.2.1 for more information about how a XML file is built.

By default the on camera look-up table is disabled. This feature is not visible until the user changes the *Visiblty* option in *Device Properties* panel to *Expert* or *Guru* (see Figure 130 and Figure 131).

In order to modify individual LUT value it is necessary that the *Visibility* option is set to *Guru*. Modifying each value in the LUT means first select the required index and second set the desired value (see Figure 131).

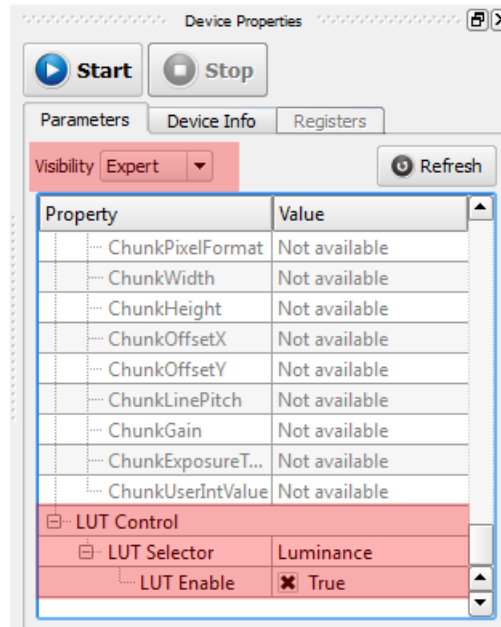


Figure 130: Enable LUT feature on Camera in GigEVisionClient

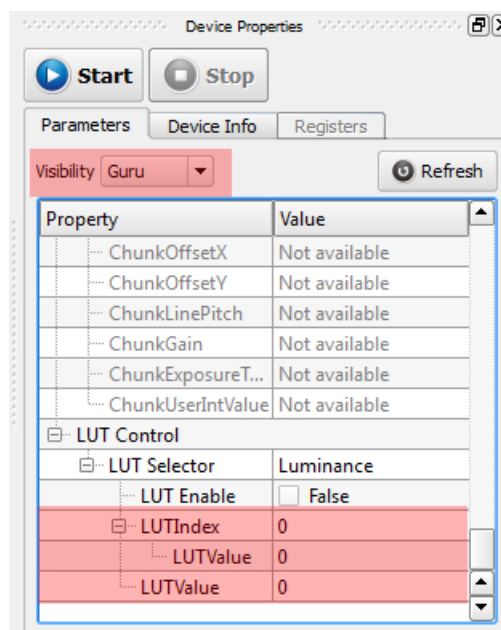


Figure 131: Modify individual LUT value in GigEVisionClient

## 6.2 Gamma Adjustment

*Gamma Adjustment* assumes that the sensor's gamma is 1.0 and comes into consideration when displaying an image on a display device. It is used to encode linear luminance to match the non-linear characteristics of display devices. Refer to chapter 7.2.5 for more details about the theory of *Gamma Correction* and/or *Gamma Adjustment*.

Gamma adjustment is realized by the following formula, where  $y'$  is the new pixel intensity,  $y$  the original pixel intensity and gamma the gamma value.

$$y' = y^{\text{gamma}}$$

Users can change the gamma value under *Analog Control* property like shown in Figure 132. The gamma value can be set in range from 0.1 to 4.0 by a step of 0.1. By default the gamma value is equal to 1.0.

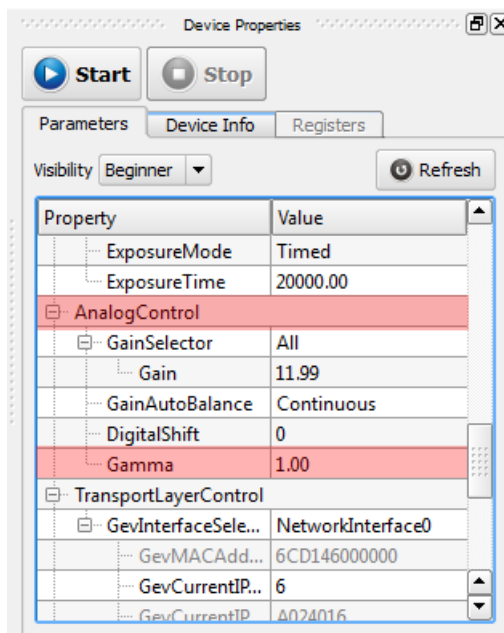


Figure 132: Modify gamma value in GigEVisionClient

## 7 Image Processing in GigEVisionSDK

The *ImageProcAPI* provided by the *GigEVisionSDK* extends the camera functionality and provides a comprehensive set of fundamental image operations and image processing algorithms, including White Balancing, Gamma Correction, Demosaicing, Color Correction and many more. All programming languages supported by the *GigEVisionSDK* API are supported by the *ImageProcAPI* as well, namely C/C++, Delphi, C# and VB .NET.

Table 86 lists all image processing algorithms implemented in the *ImageProcAPI* including the supported input image type.

Image statistics	Supported image input type		
	Monochrome	Raw Bayer	RGB
Histogram	✓	✓	✓
Average luminance	✓	✓	✓
<b>Image processing algorithms</b>			
Look-up Table (LUT)	✓	✓	✓
Digital Gain	✓	✓	✓
Auto Exposure	✓	✓	✓
White Balance		✓	✓
Gamma Correction	✓	✓	✓
Debayering / Demosaicing			
Bilinear		✓	
High-Quality Linear		✓	
Pixel Grouping		✓	
Colorized		✓	
Matrix Multiplication			✓
Gimp HSL			✓
Sharpening	✓		✓
RGB to Gray conversion			✓
Bit Depth conversion	✓	✓	✓
<b>Image Processing pipeline</b>			
Color pipeline	✓	✓	

Table 86: Implemented image processing algorithms in ImageProcAPI

The installed *ImageProcAPI* includes several code examples for reference, available in all supported programming languages. All examples are located in the corresponding folder of the *GigEVisionSDK* installation directory. For a more detailed description on the parameters of each algorithm or on how to apply them, please refer to the *GigEVisionSDK* API help located at the doc folder of the *GigEVisionSDK* installation directory.

## 7.1 Image Statistics

### 7.1.1 Histogram

A histogram is a graphical representation of the distribution of all intensity values that can be found in an image. The histogram graph is plotted using the Cartesian coordinate system. The x-coordinates are in a range from 0 to  $2^n - 1$ , where n is the bit depth of a pixel value. Each x-coordinate can correspond to an intensity value that exists in the image. The y-coordinates are in a range from 0 to  $image\_width \times image\_height$ , describing the total numbers of pixels for each pixel value found in the image.

Figure 133 illustrates a histogram of an 8-bit monochrome image (1):

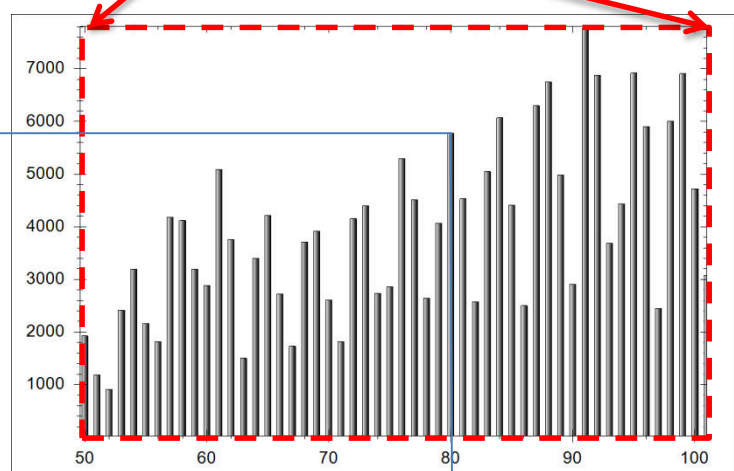
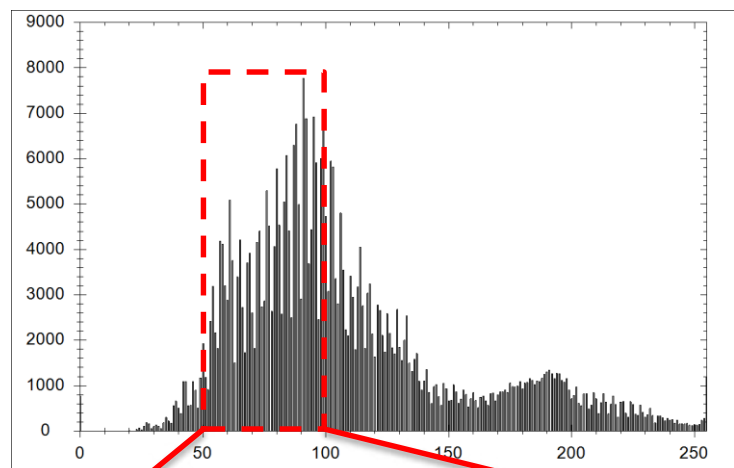
- The horizontal x-axis of the graph (2) contains the intensity values with a range from 0 to  $2^8 - 1$ , or 0 to 255. The vertical y-axis shows the count of pixels with the corresponding intensity. There are 5774 pixels that have the value 80 in the image (1) for example.
- While the graph (2) allows a quick visual interpretation of the distribution of pixels in the image, the histogram table (3) gives more exact information about the count of specific intensities. For the pixel value 80 there are exact 5774 pixels located in the image while other intensity levels, are not even present, like 1 and 2.



Pixel value	Count
0	768
1	0
2	0
...	...
80	5774
81	4535
...	...
255	1191

Histogram table

Number of pixels with  
value 80 = 5774



Pixel value = 80

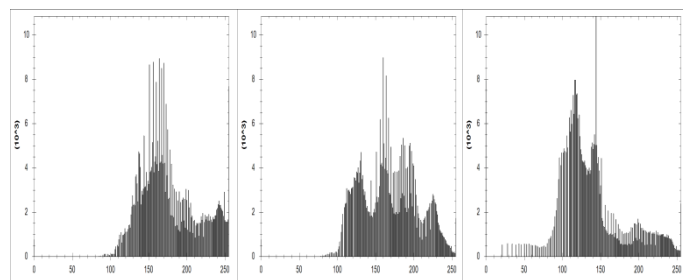
Figure 133: Source image with histogram graph and table

Histogram calculation can be applied to monochromatic or RGB image data. The number of resulting histograms corresponds to the number of channels. In the previous example there is only a single histogram for the monochrome image. However, on RGB color images are three histograms generated, each represents a single color channel (red, green and blue), illustrated in Figure 134. The application of histograms is varying; on the basis of a histogram and depending on the scene the user can for example quickly determine whether the captured image is over-, under- or normal-exposed. Figure 134 shows three color images with the corresponding histograms for each channel. Without looking to the images, the following information can be determined by the histogram:

- In the first row the population of the pixel values in each histogram is shifted to the right of the center (brighter region); it indicates an overexposed image.
- In the second row the three histograms are shifted to the left (darker region); it indicates an underexposed image.
- In the third and last row the three histograms show a centered uniform distribution of pixel intensities; it indicates an optimal exposed image.



**Overexposed Image**



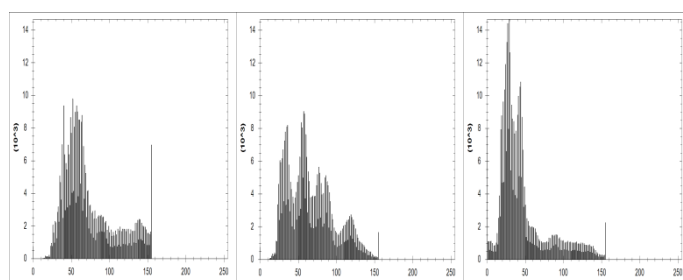
Histogram of  
red channel

Histogram of  
green channel

Histogram of  
blue channel



**Underexposed Image**



Histogram of  
red channel

Histogram of  
green channel

Histogram of  
blue channel

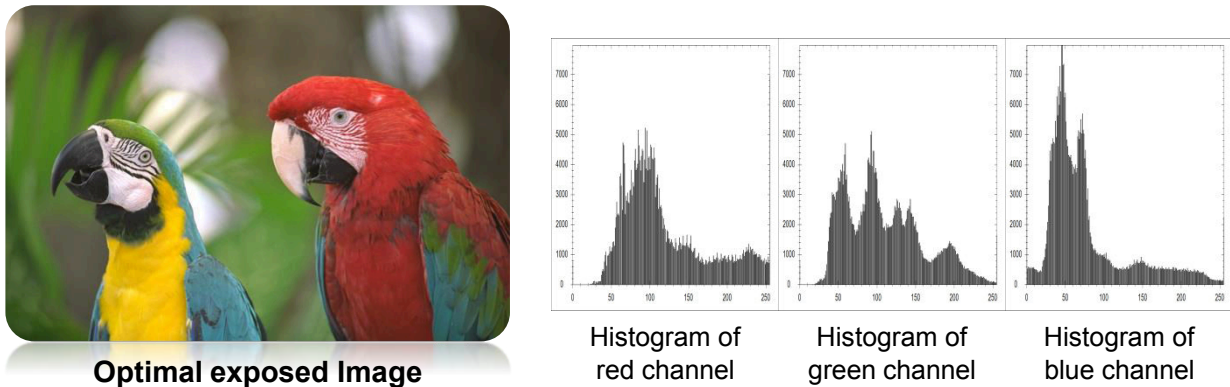


Figure 134: Example of using histogram to determine optimal image exposure

Other applications for histograms are:

- Calculation of optimal exposure time
- Calculation of correction values for white balancing
- Contrast enhancement using histogram equalization
- Global thresholding (often used in the area of image segmentation)

### Histogram in the GigEVisionSDK

In the *GigEVisionSDK* the *ImageProcAPI* provides the programming interface to generate histogram data from images. The bit depth and image types supported by the histogram feature are shown in Table 87. For a detailed description on how to instantiate this feature in user applications please refer to the *GigEVisionSDK API Help* located in the doc folder of the *GigEVisionSDK* installation directory.

Supported image input	Supported bit depth	
	8 bit per channel	16 bit per channel
Monochrome	✓	✓
Raw Bayer	✓	✓
Color RGB	✓	✓

Table 87: Histogram - supported bit depth and image types



### Histogram in the GigEVisionClient

In the *GigEVisionClient* application the histogram feature can be enabled by the menu bar entry Control ⇒ Histogram, shown in Figure 135.

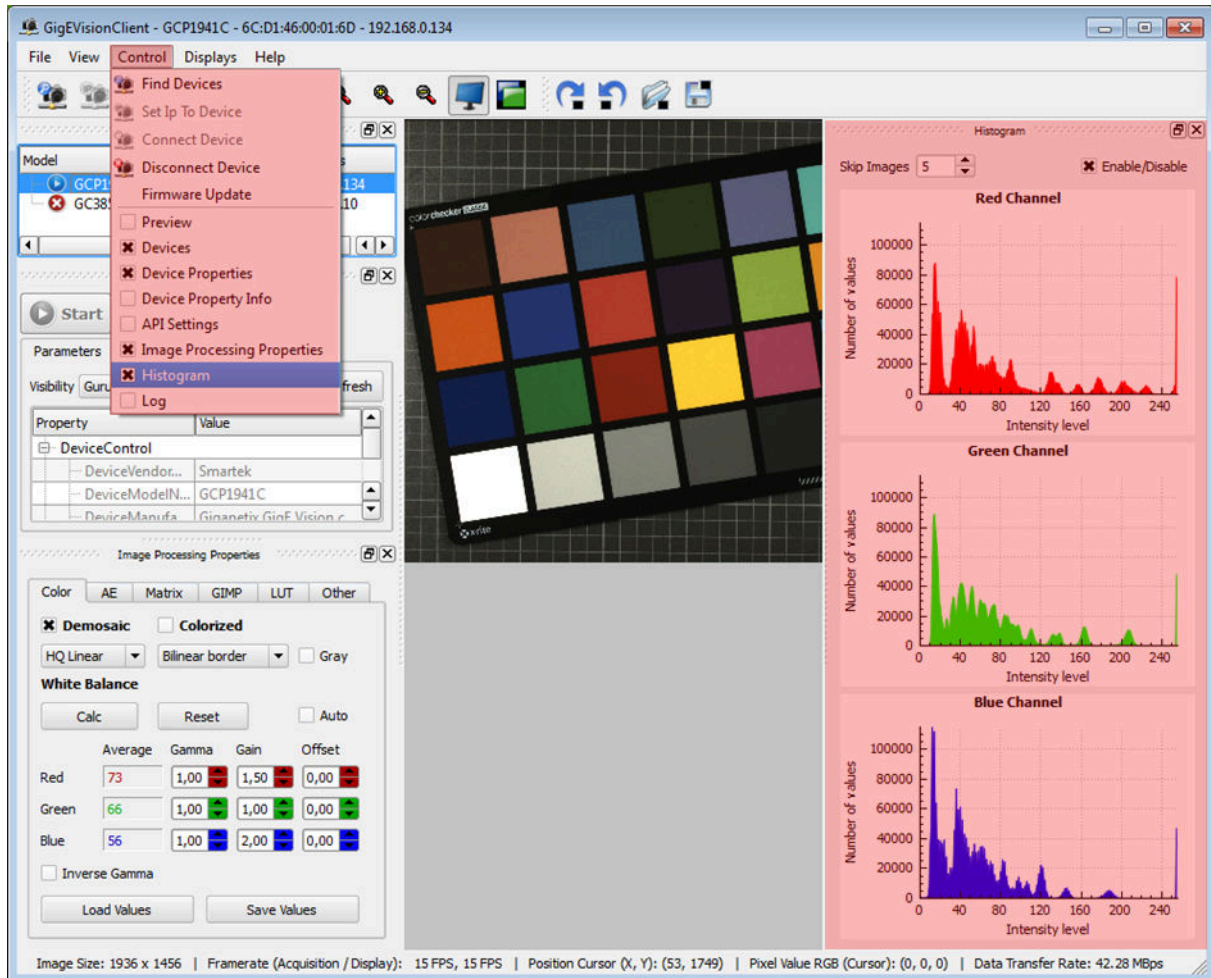


Figure 135: Histogram feature in GigEVisionClient

- **Skip Images (default 5):** Number of frames to skip before a new histogram is calculated and the GUI is updated.
- **Enable / Disable:** Active / deactivate histogram calculation.



### 7.1.2 Average Luminance Calculation

The Average Luminance algorithm sums up all pixel values on each channel of an image and calculates the average value by dividing the sum of pixel values by number of pixels on this channel:

$$\text{Averagevalue}_{\text{channel}} = \frac{\sum \text{Pixelvalue}_{\text{channel}}}{\text{Totalpixels}_{\text{channel}}}$$

For single channel images only one average value is calculated, while for raw/color images each channel has its own value.

Calculating average values of an image is a fundamental operation in image processing and serves as a basis calculation for algorithms like auto exposure and auto white balancing.

#### Average Luminance Calculation in the GigEVisionSDK

In the *GigEVisionSDK* the *ImageProcAPI* provides the programming interface to generate average luminance data of an image. The bit depth and image type supported are shown in Table 88. For a detailed description on how to use the average value calculation feature please refer to the *GigEVisionSDK API Help* located in the doc folder of the *GigEVisionSDK* installation directory.

Supported image input	Supported bit depth	
	8 bit per channel	16 bit per channel
Monochrome	✓	✓
Raw Bayer	✓	✓
Color RGB	✓	✓

Table 88: Average luminance calculation - supported bit depth and image types

### Average Luminance in the GigEVisionClient

Calculated average value(s) of the image channel(s) can be found in the *Image Processing Properties* under *Color / Mono*, shown in Figure 136. If not visible, it can be enabled by the menu bar entry *Control*  $\Rightarrow$  *Image Processing Properties*.

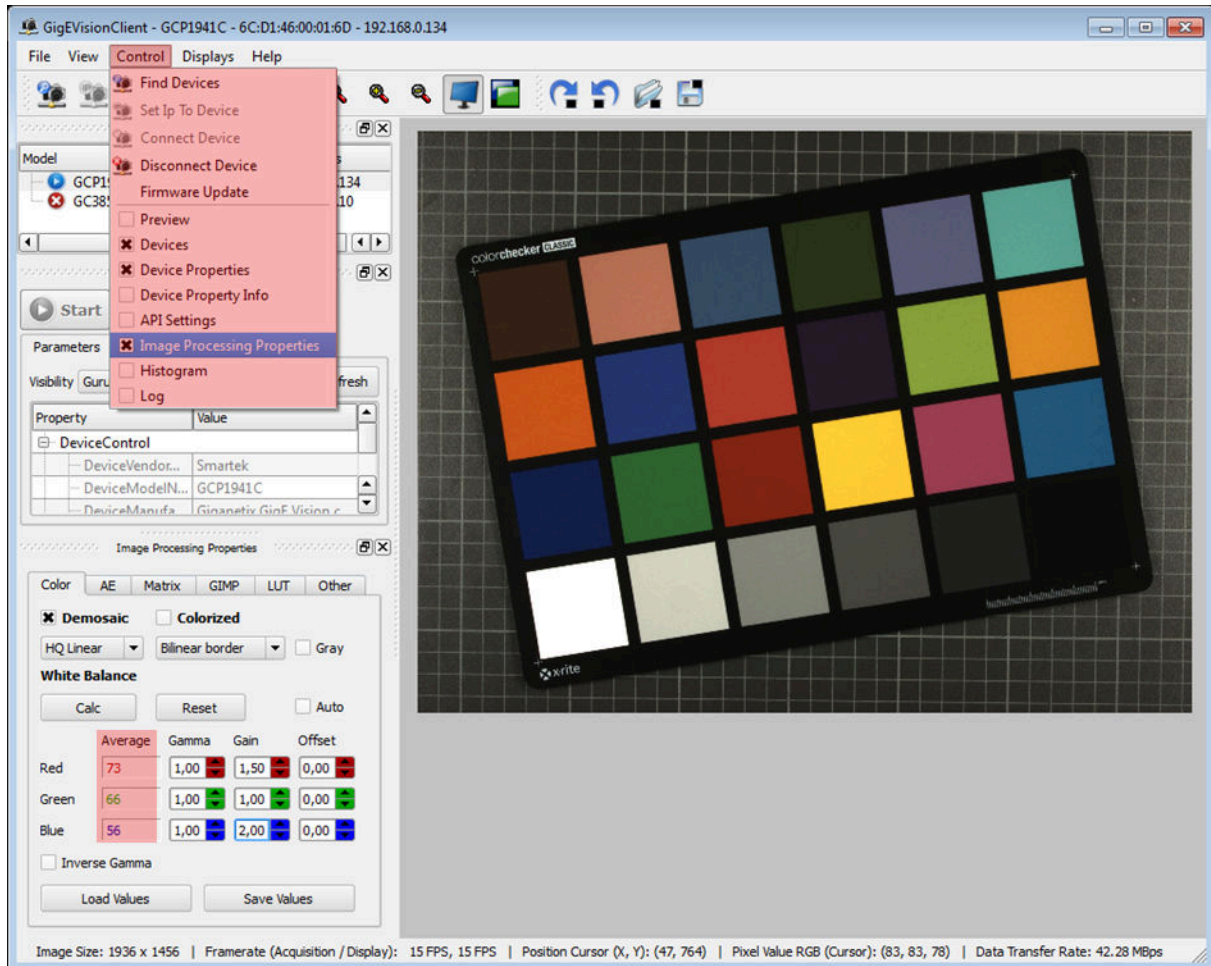


Figure 136: Average value calculation in GigEVisionClient for White Balancing algorithm

## 7.2 Image Processing Algorithms

### 7.2.1 Luminance Look-Up Table (LUT)

In image processing a Look-Up Table or LUT is used to transform input pixel values into desired output pixel values by using mapping operations.

Essentially a luminance look-up table is a list with  $2^n$  entries, where  $n$  is the bit depth of the input image. Each entry in the table has an index and represents an output luminance value. The indices are unique and numbered continuously from 0 to  $2^n - 1$ . Figure 137 shows an example of an 8 bit look-up table.

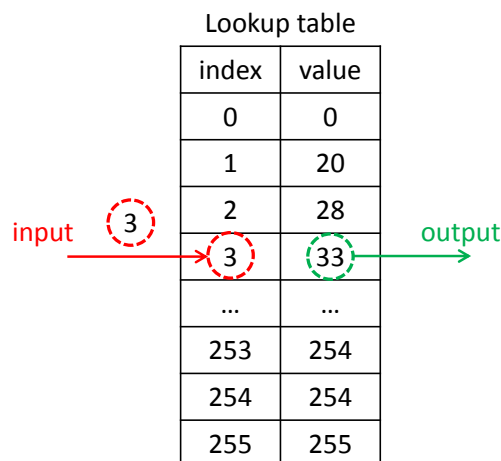


Figure 137: 8 bit look-up table

The index represents the luminance of a pixel in the image, which is exchanged by the value:

- When an input pixel value has a value of 1, this value will be used as the index in the table. The corresponding 8 bits output pixel value at index 1 will be 20.
- When an input pixel value has a value of 3, this value will be used as the index in the table. The corresponding 8 bits output pixel value at index 3 will be 33.
- When an input pixel value has a value of 253, this value will be used as the index in the table. The corresponding 8 bits output pixel value at index 253 will be 254.

Look-up tables are especially suited for point operations in image processing where output pixel values depend only on the corresponding input pixel values. In the following a couple of application examples using look-up tables are shown.

The first example shows a look-up table where each output pixel value is mapped to its exactly corresponding input pixel value. The look-up table is a linear function ( $f(x) = x$ ) and its graph is a  $45^\circ$  straight line, shown in Figure 138. Because of the one-to-one value mapping the output image is identical to the input image.

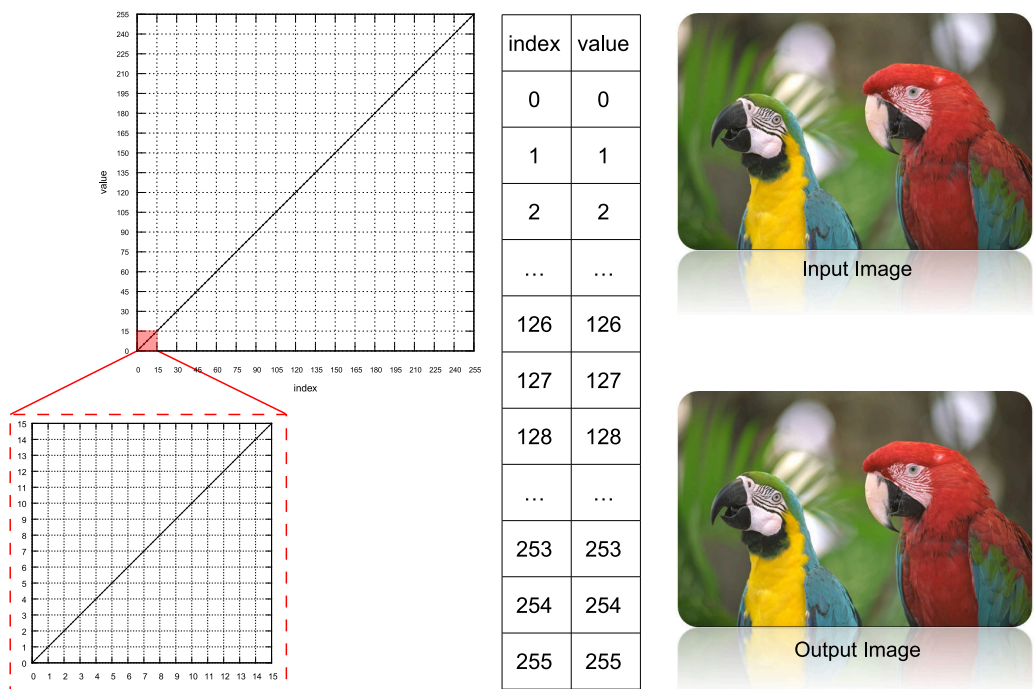


Figure 138: Linear transformation using 8 bit look-up table

The second example demonstrates a variant of gamma correction using a look up table. By reference to the look-up table and its corresponding graph, in Figure 139, it is visible that a non-linear transformation is applied to the input pixel values.

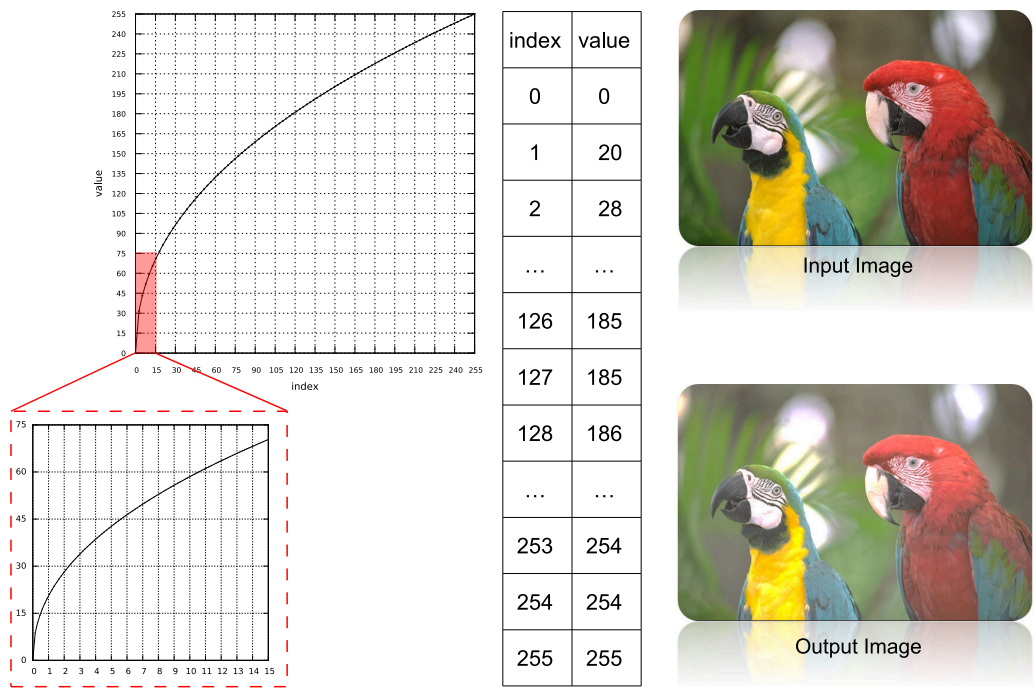


Figure 139: Gamma correction using look-up table

The third example, illustrated in Figure 140, shows the inverting of an 8 bit monochrome image by a LUT. Every input gray level value is transformed into an output gray-level value by the formula  $\text{Value}_{\text{out}} = 2^8 - \text{Value}_{\text{in}}$ .

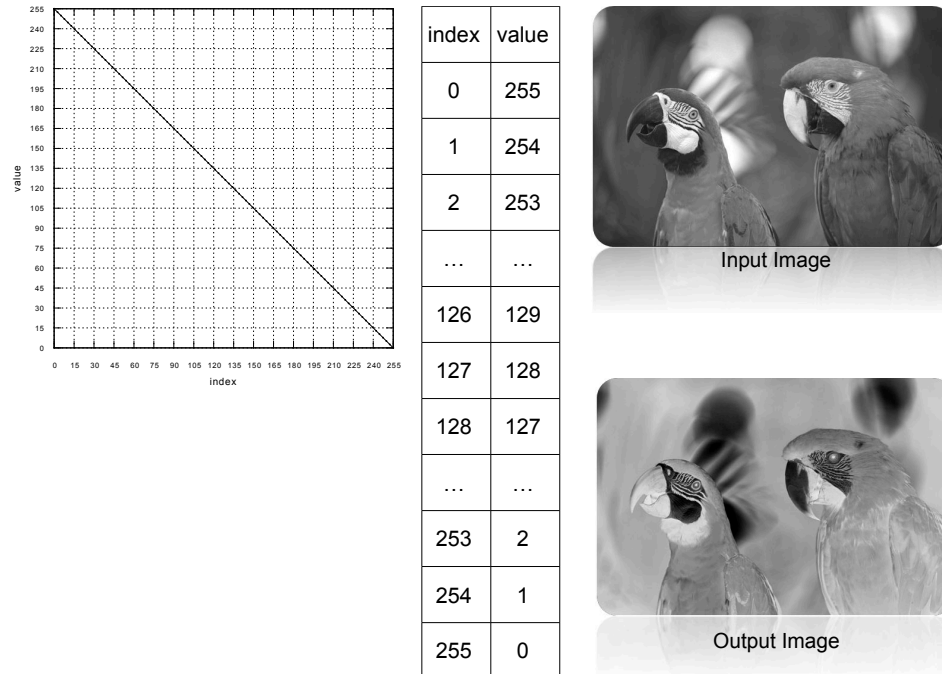


Figure 140: Inverting a monochrome image using look-up table

The last example demonstrates two implementations of contrast enhancement, using a look-up table applied to an 8-bit per channel color image. In Figure 141 the first 45 pixel values have been set to 0 and pixel values in range from 173 to 255 have been set to 255.

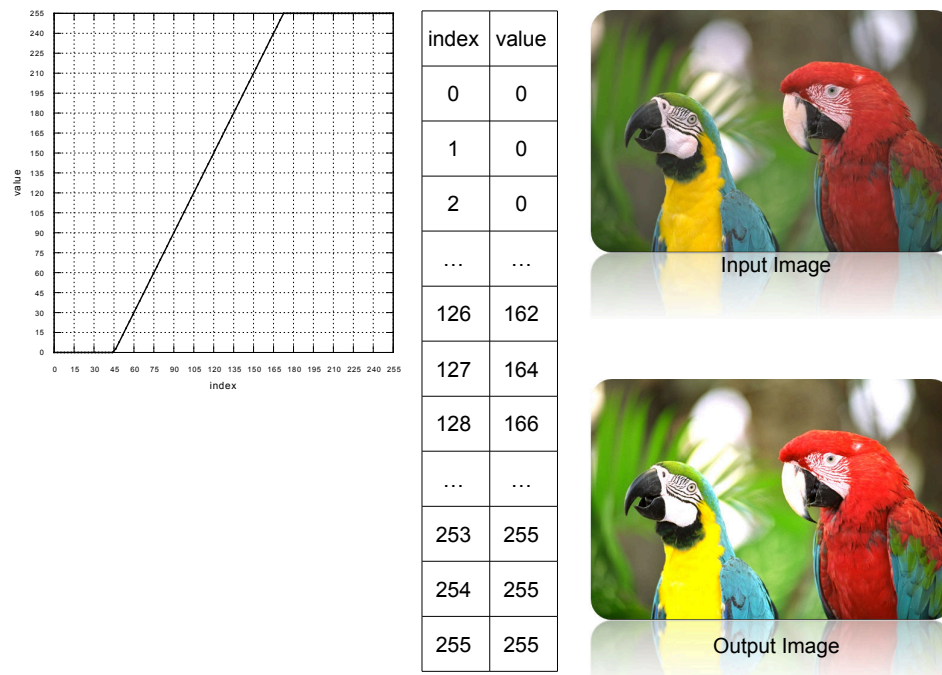


Figure 141: Enhancing contrast of an image using look-up table



Figure 142 shows the same purpose by a complex function and illustrates that the implemented algorithms can be arbitrarily complex. However, the calculation for generating look-up tables will be executed only once.

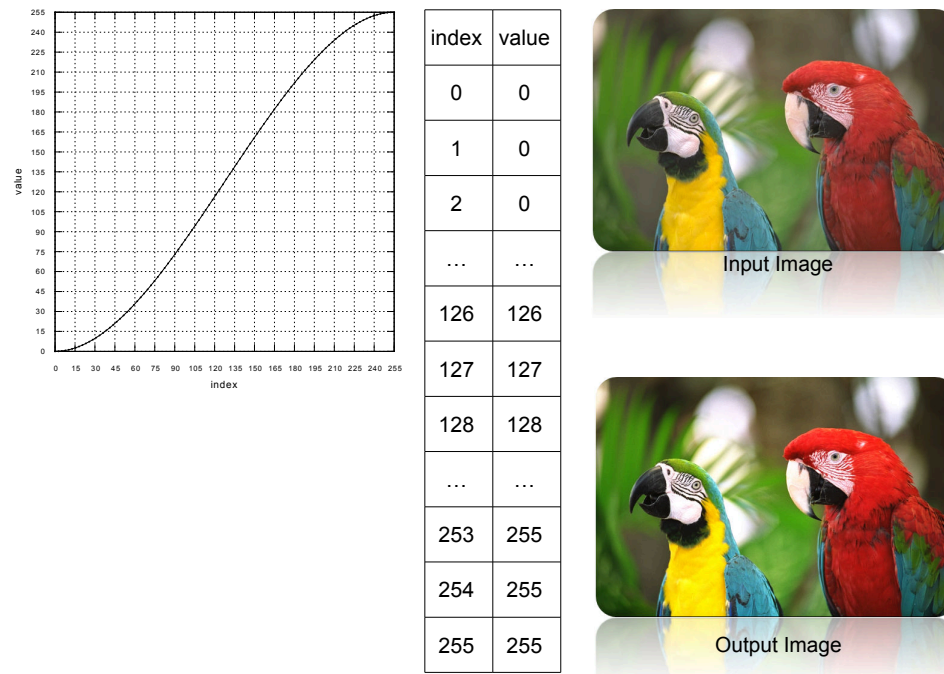


Figure 142: Example 2 of enhancing contrast of an image using look-up table

### Look-up table in GigEVisionSDK

In the *GigEVisionSDK* the *ImageProcAPI* provides the programming interface for generating and modifying look-up tables. The bit depth and image type supported are shown in Table 89. For a detailed description on how to use the look-up table feature please refer to the *GigEVisionSDK API Help* located in the doc folder of the *GigEVisionSDK* installation directory.

Supported image input	Supported bit depth	
	8 bit per channel	16 bit per channel
Monochrome	✓	✓
Raw Bayer	✓	✓
Color RGB	✓	✓

Table 89: Look-up table - supported bit depth and image types

### Look-up table in GigEVisionClient

The *GigEVisionClient* provides the user the ability to load a pre-defined look-up table in form of an *XML* file into the application. XML examples for look-up tables are located in the *GigEVisionSDK* installation folder:

\$(GIGE\_VISION\_SDK\_PATH)\GenICam\_v2\_4\xml\custom\

```
<?xml version="1.0" encoding="UTF-8"?>
<values>
  <color channel="Red">
    <LUT index="0" value="230"/>
    <LUT index="1" value="57"/>
    <LUT index="2" value="28"/>
    ...
    <LUT index="254" value="72"/>
    <LUT index="255" value="67"/>
  </color>
  <color channel="Green">
    <LUT index="0" value="208"/>
    <LUT index="1" value="96"/>
    <LUT index="2" value="253"/>
    ...
    <LUT index="254" value="231"/>
    <LUT index="255" value="42"/>
  </color>
  <color channel="Blue">
    <LUT index="0" value="206"/>
    <LUT index="1" value="74"/>
    <LUT index="2" value="146"/>
    ...
    <LUT index="254" value="250"/>
    <LUT index="255" value="182"/>
  </color>
</values>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<values>
  <color channel="Greyscale">
    <LUT index="0" value="230"/>
    <LUT index="1" value="57"/>
    <LUT index="2" value="28"/>
    ...
    <LUT index="254" value="72"/>
    <LUT index="255" value="67"/>
  </color>
</values>
```

Figure 143: User-defined XML file for 8 Bit RGB color (left) or monochrome (right) images

Figure 143 shows an example of a properly formatted XML file which contains look-up table parameters for 8-bit per channel color and monochrome images. The first line indicates the root element values. Element color with attribute channel indicates the channel for which the parameters will be set. The Child element LUT with the attribute index indicates the index or input value, the attribute value indicates the output value for the current index.

Figure 144 shows the look-up table feature located in the *LUT* tab within the *Image Processing Properties* panel. If not visible, the *Image Processing Properties* panel can be activated by the menu bar entry *Control* ⇒ *Image Processing Properties*.

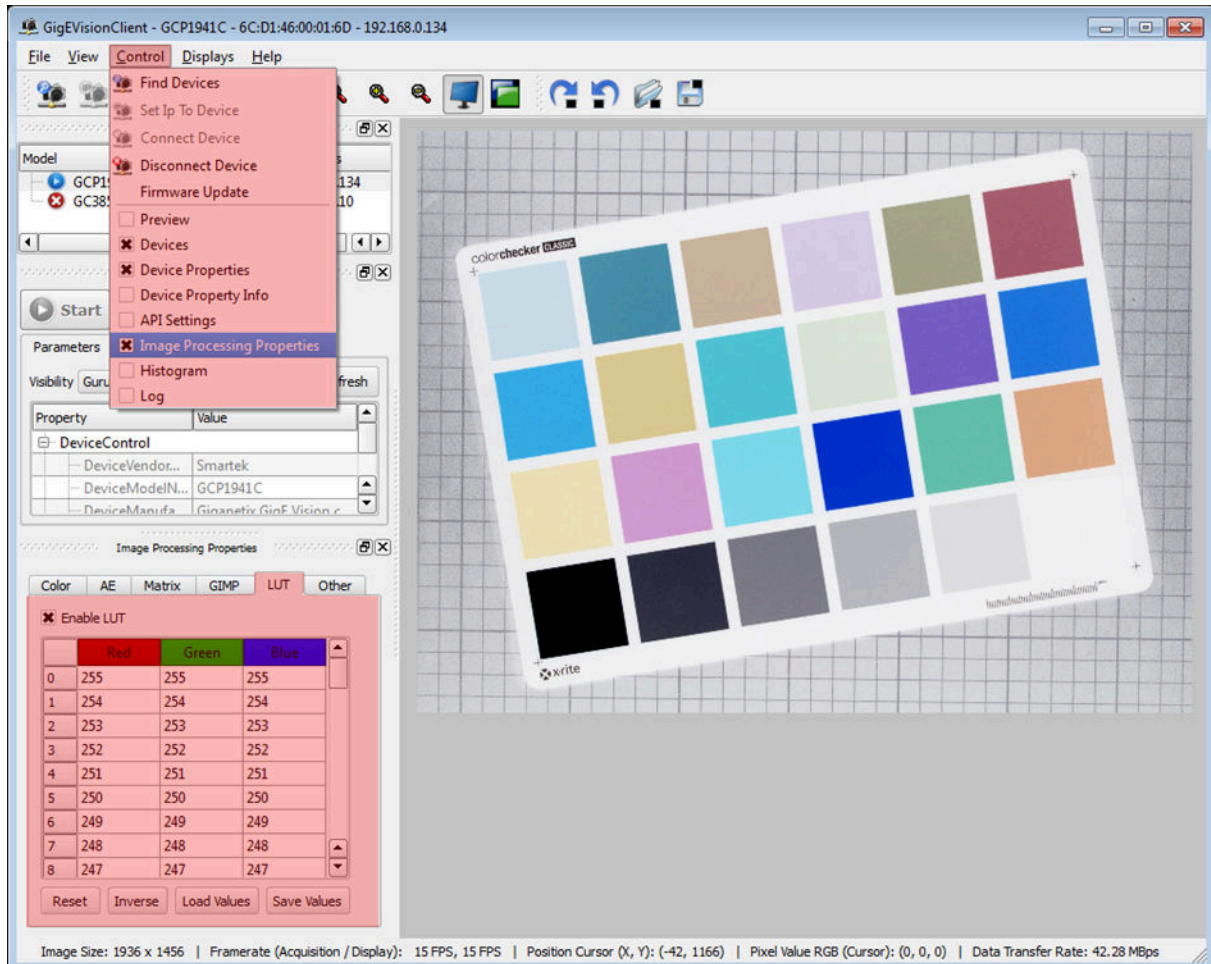


Figure 144: Look-up table feature in GigE Vision Client

- **Enable LUT:** Enable application of look-up table
- **Reset:** Reset look-up table to default values
- **Load Values:** Load an user-defined XML file with look-up table parameters into the client
- **Save Values:** Save the user-defined look-up table to a file
- **Inverse:** Generate a predefined look-up table which inverts the image



### 7.2.2 Digital Gain

The pixel signal received from an image sensor is amplified and digitized before transmitted to the host application. For devices which do not provide an individual analog gain separately for each color channel, or in applications where the available maximum analog gain does not suffice, a software based gain can be applied by the *ImageProcAPI*. The digital gain is a factor which is multiplied with each pixel value of an image channel, generating the new value of the pixel:

$$\text{Pixel}(x, y)_{\text{out}} = \text{Pixel}(x, y)_{\text{in}} \times \text{DigitalGain}$$

Each channel has its own gain value, which makes it for example to a tool for white balancing, if not already supported by the camera.

Further, digital gain is a useful feature to enhance the image brightness, especially under low light condition. Increasing a digital gain value means increasing the intensity of each pixel, resulting in a brighter overall image. However, the image noise will also be increase with digital gain.

Figure 145 demonstrates four different gain settings applied to the image. While digital gain equals 1.0 represents the image at its original, with increasing digital gain value, the image becomes brighter and the noise rises as well. Also at higher gain settings, some pixels are over-saturated what leads to information loss in the image.



Digital Gain = 1.0



Digital Gain = 2.0



Digital Gain = 3.0



Digital Gain = 4.0

Figure 145: Digital Gain to brighten an image



#### Note

In contrast to the analog gain the digital gain produces "holes" in the histogram, shown in Figure 146. As the multiplication takes place on the digitized image with the same bit depth as the output image, some luminance levels cannot be reached anymore.

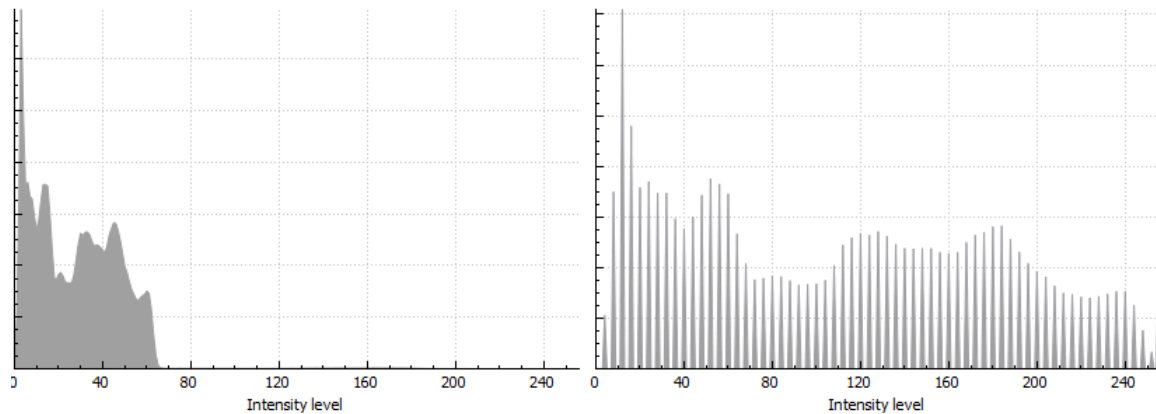


Figure 146: Digital Gain - Histogram original image (left) and after applying digital gain of 4.0 (right)

With a digital gain of 2.0 it is for example not possible to receive any uneven values (1; 3; 5...), like sketched in Table 90. The analog gain is therefore always to be preferred where possible.

Pixel <sub>In</sub>	Pixel <sub>Out</sub> = Pixel <sub>In</sub> × 2.0
0	0
1	2
2	4
3	6

Table 90: Digital Gain - Output values

### Digital Gain in GigEVisionSDK

In the *GigEVisionSDK* the *ImageProcAPI* provides the programming interface to apply digital gain to images. The bit depths and image types supported are shown in Table 91. For a detailed description on how to use the digital gain feature please refer to the *GigEVisionSDK API Help* located in the doc folder of the *GigEVisionSDK* installation directory.

Supported image input	Supported bit depth	
	8 bit per channel	16 bit per channel
Monochrome	✓	✓
Raw Bayer	✓	✓
Color RGB	✓	✓

Table 91: Digital Gain - supported bit depth and image type

### Digital Gain in the GigE VisionClient

In the *GigE VisionClient* the *Digital Gain* can be accessed in the *Image Processing Properties* panel under *Color / Mono*, shown in Figure 147. If not visible, the panel can be enabled by the menu bar entry *Control* ⇒ *Image Processing Properties*.

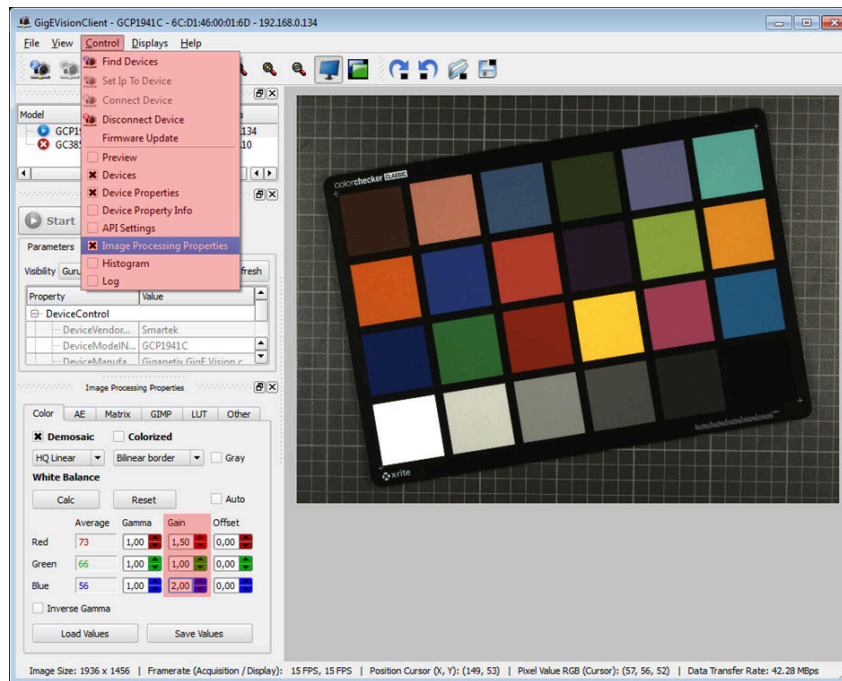


Figure 147: Digital Gain in GigE VisionClient



#### Note

The Digital Gain is used to apply the White Balancing values to the image. While the "Auto White Balance" option is enabled, a manual configuration of digital gain is not possible.

### 7.2.3 Auto Exposure and Auto Gain

Cameras are often used in different environments and applications with changing conditions, what also includes the illumination situation which may vary and change constantly. The exposure time determines how bright or dark an image will appear; the longer the exposure, the brighter the image and vice versa. The automatic exposure feature of the *ImageProcAPI* will automatically adjust the exposure time of SMARTEK Vision cameras within defined limits, until the specified target brightness is reached.

Increasing the exposure time means decreasing the maximum possible frame rate, therefore in various applications, where a specific minimum frame rate is required, the exposure time may not be arbitrarily high. In this situation the brightness can be further increased applying a digital gain. The Auto Exposure feature in the *ImageProcAPI* provides therefore a property to limit the maximum allowed exposure time, from which the gain will be increased instead.

#### Auto Exposure in the GigEVisionSDK

In the *GigEVisionSDK* the *ImageProcAPI* provides the programming interface to set parameters and execute the auto exposure algorithm to determine the new exposure time and gain adjustment values. The bit depth and image type supported are shown in Table 92. For a detailed description on how to use the auto exposure feature please refer to the *GigEVisionSDK API Help* located in the doc folder of the *GigEVisionSDK* installation directory.

Supported image input	Supported bit depth	
	8 bit per channel	16 bit per channel
Monochrome	✓	✓
Raw Bayer	✓	✓
Color RGB	✓	✓

Table 92: Auto exposure - supported bit depth and image type

### Auto Exposure in the GigEVisionClient

In the *GigEVisionClient* the *Auto Exposure* (AE) can be enabled / disabled in the *Image Processing Properties* panel under *AE* (see Figure 148). If not visible, the panel can be enabled by the menu bar entry *Control*  $\Rightarrow$  *Image Processing Properties*.

Four parameters can be adjusted:

1. **Target Brightness [%] (default 50):** This parameter determines the average brightness of the image which should be reached. For an 8-bit image this value is 127.5, for a 16-bit image 32767.5.
2. **Min Exposure Time [ $\mu$ s] (default 100):** minimum exposure time to be calculated. This value must not match the minimum exposure time of the image sensor, but should not undercut.
3. **Max Exposure Time [ $\mu$ s] (default 300000):** maximum exposure time to be calculated. This value must not match the maximum exposure time of the camera, but should not exceed.
4. **Exposure Time Threshold [%] (default 10):** absolute difference between new exposure and old exposure value. The new calculated exposure value needs to be higher than this threshold value to be considered as the new exposure to be adjusted.

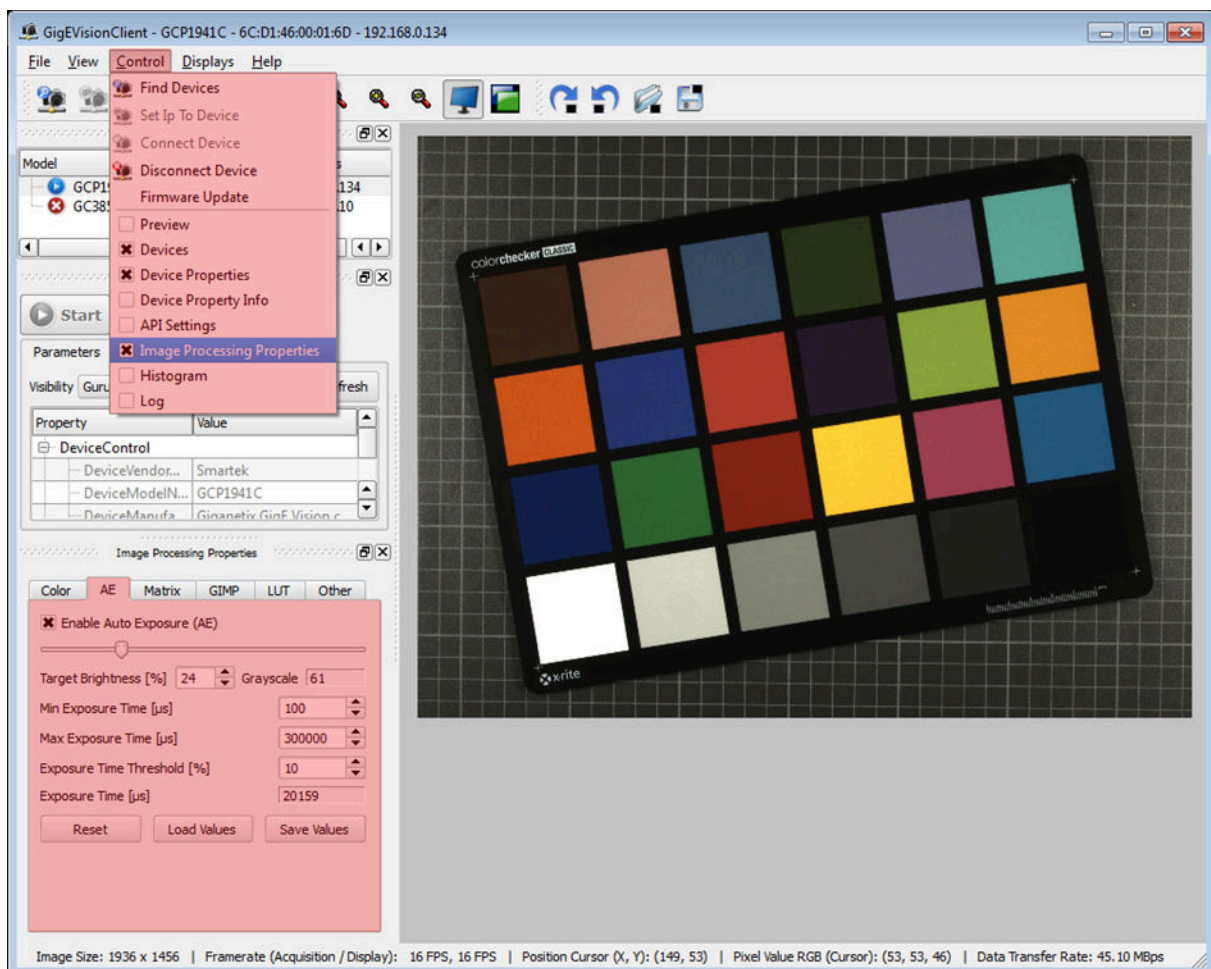


Figure 148: Auto Exposure in GigEVisionClient.

### 7.2.4 White Balance

White Balancing is the process of removing unwanted color casts in digital images, which derived from one important characteristic of visible light - the color temperature. The color temperature is defined by the radiation emitted by a glowing "black body" and is measured in Kelvin (K). Since an image sensor converts light to electrical voltage which then underruns multiple processing steps until a digital image is saved or displayed on the screen, the color temperature of the light is visible on the digital image in form of color casts appearing to the human eye.

Figure 149 illustrates the color temperature spectrum of visible light in the range from 1000K to 15000K.

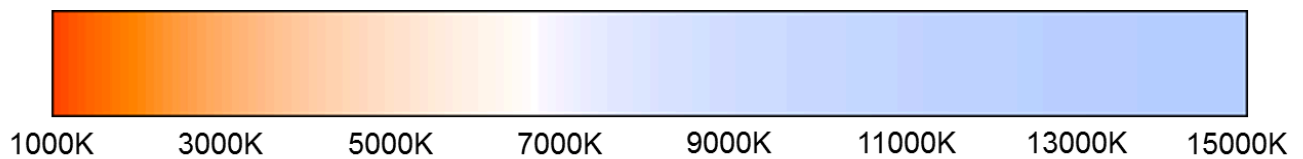


Figure 149: Color temperature spectrum in range 1000K to 15000K

Digital images which are captured in lower color temperature conditions (candle light, tungsten light) tends to be reddish or reddish orange. The higher the color temperature (overcast sky) the more blue light will outweigh, the digital image appears more bluish.

To fully describe color casts in digital images a tint adjustment is also required. While the color temperature determines the warmth or coolness of an image, the tint defines the balance between magenta and green color casts.

Figure 150 shows two images of a color checker chart. The image in the left shows the original values of the patches while the color checker on the right is captured by a camera at day light condition. If we look at the last row of the color chart on the left image, the Gray color fields tend to be green.



Figure 150: Comparison original color (left) and camera image (right) of a ColorChecker chart

Human eyes can automatically correct this effect, for a camera to compensate this effect automatic white balancing is needed to accurately balance color. The white balancing feature implemented in the *ImageProcAPI* adjusts the weighting for each color channel using digital gains in order to remove the unwanted color casts.



Figure 151 demonstrates the White Balancing feature implemented in the *GigEVisionSDK*. The green color cast is corrected, the output image appears as it should.



Figure 151: ColorChecker chart without (left) and with White Balancing (right)

### White Balance in the GigEVisionSDK

In the *GigEVisionSDK* the *ImageProcAPI* provides the programming interface for executing the White Balance algorithm. The bit depth and image types supported are shown in Table 93

Supported image input	Supported bit depth	
	8 bit per channel	16 bit per channel
Monochrome	✓	✓
Raw Bayer	✓	✓
Color RGB	✓	✓

Table 93: White Balance - supported bit depth and supported image type

For a detailed description on how to use the auto white balance feature please refer to the *GigEVisionSDK API Help* located in the doc folder of the *GigEVisionSDK* installation directory.

### White Balance in the GigEVisionClient

In *GigEVisionClient* the user can apply the *White Balance* algorithm once or repeatedly for every incoming frame. All options can be accessed in the *Image Processing Properties* panel under *Color* shown in Figure 152. If not visible, the panel can be enabled by the menu bar entry *Control* ⇒ *Image Processing Properties*.

The single white balance mode is recommended in scenes where the lighting condition is constant, so there will be no computation overhead. The correction values are calculated once when the *Calc* button is pressed.

The Auto White Balance mode is disabled by default, as soon as enabled by the *Auto White Balance (AWB)* check box it calculates and applies correction gains for every frame. This mode is recommended when the lighting condition may permanently change.

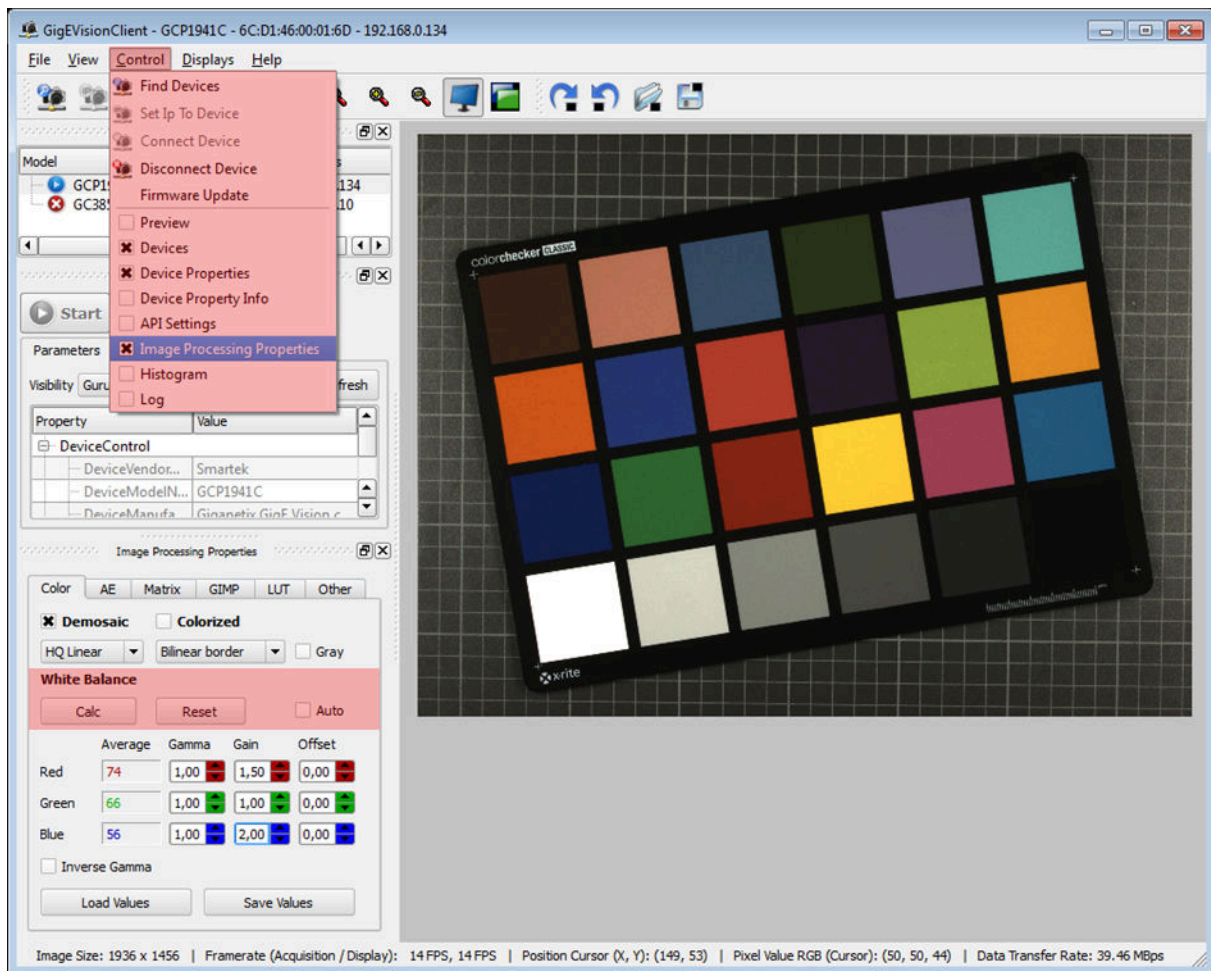


Figure 152: White Balance feature in the GigEVisionClient

- **Calc:** Start white balancing calculation once.
- **Auto:** Repeatedly apply white balancing to the images.
- **Reset:** Reset every results calculated by the white balancing process to default. If auto white balance is enabled before, it will be disabled.



### 7.2.5 Gamma Correction

Gamma is an important characteristic of digital imaging systems, as it translates between the light sensitivity of the human eye and thus of the image sensor. Generally it has to be distinguished that there are basically two definitions of gamma. The first one is called *Gamma Correction*, the second one *Gamma Adjustment*. The *Gamma Adjustment* assumes that the sensor's gamma is 1.0 and comes into consideration when displaying an image on a display device. It is used to encode linear luminance or RGB values to match the non-linear characteristics of display devices.

Depending on the characteristics of the sensor and also influenced by gain or black level, the gamma output of the camera is not ideally 1.0. If only the Gamma Adjustment is applied to the image, the real gamma may represent a combination of the encoding gamma and the sensor's gamma. The consequence of this effect is that the brightness levels of the image outputted on the display are distorted.

In situations where the gamma of the sensor is not 1.0, the *Gamma Correction* can be used to linearize the non-linear sensor's characteristics to match a linear gamma of 1.0. For this purpose a well calibrated gray scale is usually used to determine the *Gamma Correction* values. The gamma value can be applied using the *ImageProcAPI*.

The term *Gamma Correction* will be used throughout this document and depends on the context, it can be understood as either *Gamma Correction* or *Gamma Adjustment*.

#### Gamma Correction Workflow

The workflow of correcting gamma is illustrated in Figure 153. First an image of a known and calibrated object like a Color Checker chart will be captured. Based on this the gamma of the sensor can be determined, a sample of a sensor gamma curve is shown in the first block of Figure 153 (from left to right). After applying the *Gamma Correction* value the brightness levels should match the values known from the calibration chart and represent a Gamma of 1.0.

If it is intended to display the image to the human eye the gamma of the display device should be known. Based on the device's gamma the *Gamma Adjustment* process could be started, encoding linear luminance to match the non-linear characteristics of the display device; a common display gamma value is 2.2. After gamma adjustment the displayed image appears luminance correct on the display device.

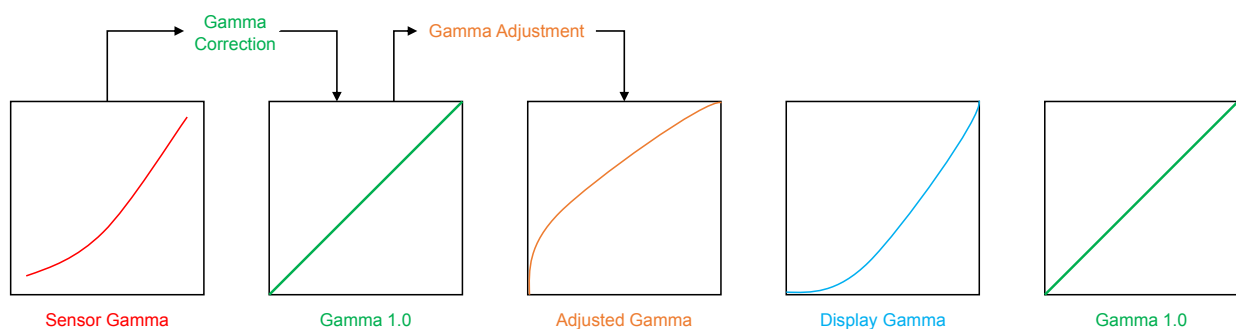


Figure 153: Gamma correction workflow

Gamma curves are defined by the following formula, where  $x$  is the percentage input luminance,  $y$  the percentage output luminance,  $a$  the darkest percentage brightness value (ideally 0),  $b$  the digital gain and  $c$  the gamma.

$$y = a + bx^c$$

Further  $x$  can be determined based on the bit depth  $n$  by the following formula:

$$x = \frac{\text{pixelvalue}}{2^n - 1}$$

The appropriate gamma curve can be determined by capturing an image of a calibrated gray scale showing a linear progression over a brightness of 0% to 100%.

### Gamma Correction in GigEVisionSDK

In the *GigEVisionSDK* the *ImageProcAPI* provides the programming interface for setting and executing the gamma correction algorithm. The bit depth and image types supported are shown in Table 94. For a detailed description on how to use the digital offset, gain and gamma correction feature please refer to the *GigEVisionSDK API Help* located in the doc folder of the *GigEVisionSDK* installation directory.

Supported image input	Supported bit depth	
	8 bit per channel	16 bit per channel
Monochrome	✓	✓
Raw Bayer	✓	✓
Color RGB	✓	✓

Table 94: Gamma Correction - supported bit depth and image type

### Gamma Correction in GigEVisionClient

In the *GigEVisionClient* Gamma, Gain and Offset can be accessed in the *Image Processing Properties* panel under *Color / Mono*, shown in Figure 154. If not visible, the panel can be enabled by the menu bar entry *Control*  $\Rightarrow$  *Image Processing Properties*.

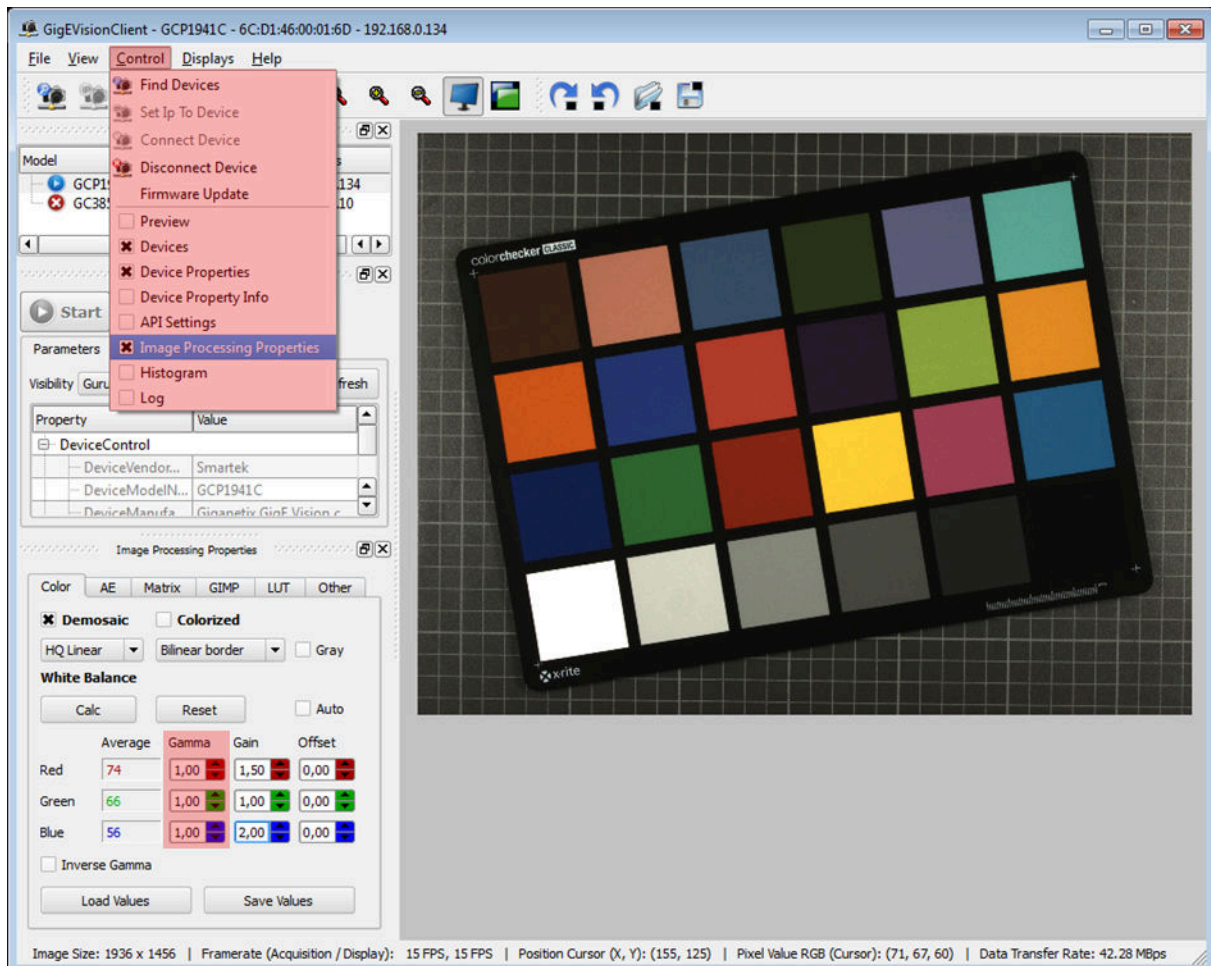


Figure 154: Gamma Correction dialog

### 7.2.6 Color Filter Array Interpolation (Demosaicing / Debayering)

Each pixel on a digital camera sensor contains a light sensitive photo diode which measures the amount of incoming light. As photodiodes are monochromatic devices, they are unable to determine the distribution of the incoming light to different wavelengths. A common way to distinguish between different light wavelengths or colors is to place an array of color filters (*Color Filter Array*; CFA) on top of the sensor to filter out for example the red, green, and blue components of light falling onto it. Among many CFA patterns, the most commonly used is the Bayer pattern. For each  $2 \times 2$  set of pixels, two diagonally opposed pixels are equipped with filters which are only transmissive for green, the other two only for red and blue. Since green carries most of the luminance information for the human eye, its sampling rate is twice as that of R and B. Figure 155 shows the "GR" filter alignment, which means that the pattern starts with green (G) followed by red (R).

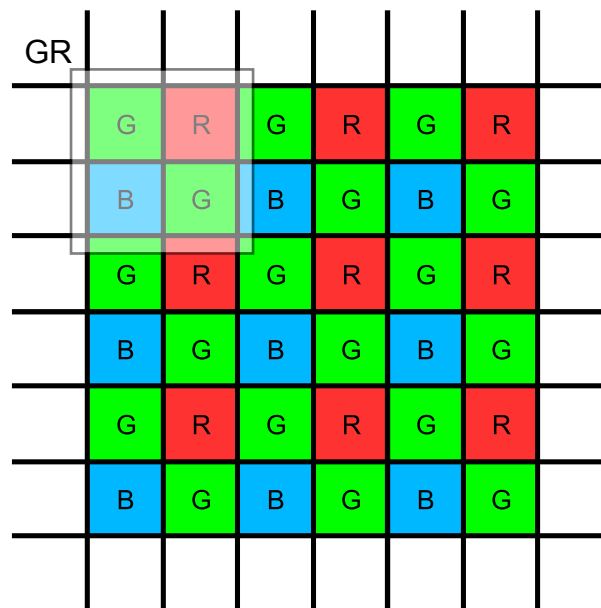


Figure 155: Bayer Filter Pattern GR

The illustration in Figure 155 is for demonstration purposes only. In effect, each pixel is described by an intensity value, which appears gray to the human eye, shown in Figure 156.

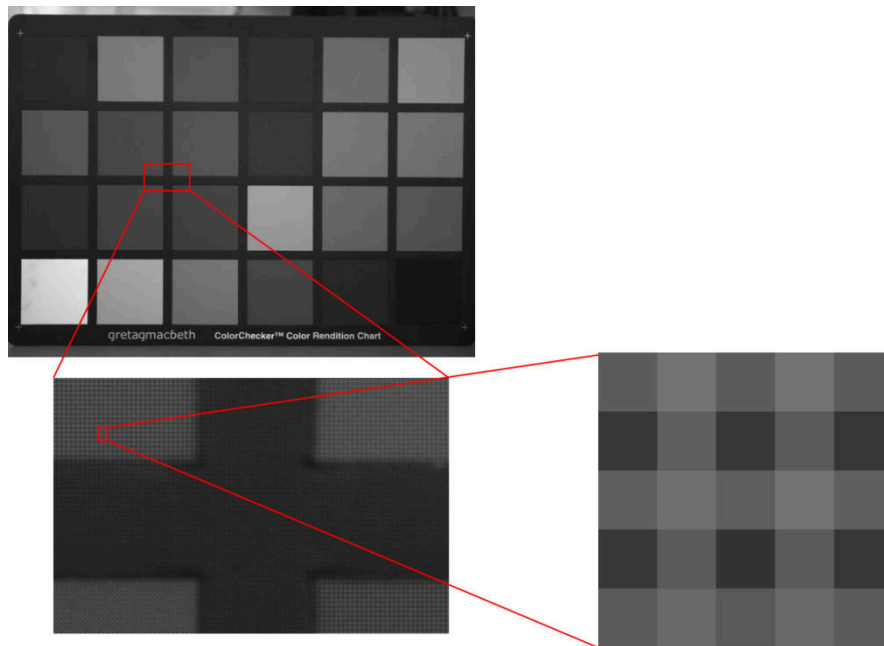


Figure 156: Raw image overlaid with a Bayer pattern

Figure 156 shows a raw image from a color camera. If it is zoomed into the image, the Bayer pattern gets more and more visible. Each pixel represents an intensity value, to reconstruct a full color image from the incomplete color samples, the missing color information at each pixel has to be interpolated. The interpolation process has several different names like *Color Filter Array Interpolation*, *Demosaicing* or *Debayering*. The reconstructed image is typically accurate in uniform-colored areas, but has a loss of resolution especially at structures and edges.

There are different interpolation methods where each of them has its own strengths and weaknesses. In the *ImageProcAPI* three algorithms are implemented, namely *Bilinear Interpolation*, *High Quality Linear Interpolation* and *Pixel Group Interpolation*.

### Bilinear Interpolation

The *Bilinear Interpolation* is a linear demosaicing method using a 3-by-3 filter for color interpolation. For each pixel its 8 direct neighbors are considered to determine the 2 missing colors of this pixel by simple averaging. The red value of a non-red pixel is computed as the average of the two or four adjacent red pixels, and similarly for blue and green.

The bilinear method has the lowest complexity as there are only a few calculations per pixel, compared to the other algorithms. It thus shows the lowest workload, but is much more imprecise at e.g. structures and edges in the image. Because of the small amount of calculations, also the memory usage is negligible compared to HQ Linear Interpolation.

### HQ Linear Interpolation

The *HQ Linear interpolation* is a gradient-corrected bilinear interpolated method using a 5x5 linear filter. In contrast to the bilinear method, the HQ Linear interpolation correlates different color channels to calculate the missing color value. For example, the red value of a non-red pixel is computed as the average of the two or four adjacent red pixels (depending on the amount of red pixels in the 5x5 neighborhood) plus a correction value calculated from pixels of a different color channel.

In comparison with the bilinear interpolation, the *HQ Linear interpolation* method has the modest increase in computational complexity. However, the main advantage of this method is that it generates significant higher quality color images with greatly reduced edge artifacts. Therefore *HQ Linear interpolation* is in the *GigEVisionSDK* used as the standard demosaicing algorithm.

### Pixel Group Interpolation

*Pixel Grouping* is another interpolation method considering pixel values in a 5x5 neighborhood to calculate missing color values. It basically works in two phases; first it computes all the unknown green values, and then it uses the input data along with the green values computed in the first phase, to compute all the missing red and blue values. The main principle is to determine the gradients in the four directions from the current processed pixel and select the value with the smallest one for final calculation. The smallest gradient value is chosen to reduce edge artifacts due to the fact that a higher gradient value is an indication for edge transition.

In comparison with the *bilinear* and *HQ Linear interpolation* methods, *Pixel Grouping* is the most memory and computational intensive algorithm. However, the result color image is at very high quality with very little edge artifacts, especially for scenes with large areas of uniform colors that are separated by clear boundaries.

### Colorized Output

The Colorized algorithm is not doing any interpolation. It simply creates an intensity Bayer RGB color image by setting the missing color values to zero. The intensity value of the current pixel remains unchanged.

### Restrictions at the Image Borders

Nearly all interpolation methods have problems at the borders of the image. Depending on the size of the filter used (3x3, 5x5, ...), one or more neighbors in each direction are needed for interpolation; at the borders at least one direction is not available, like illustrated in Figure 157 and Figure 158 for *Bilinear* and *HQ Linear*.

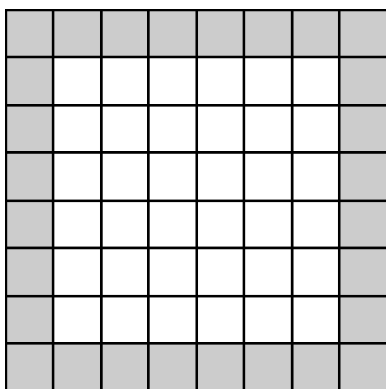


Figure 157: Bilinear algorithm border

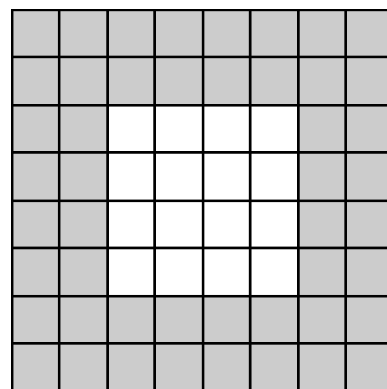


Figure 158: HQ Linear algorithm border

The ImageProcAPI therefore provides four approaches:

- leave border pixels as original (RAW)
- cut off border pixels
- fill border pixels with a solid color
- interpolate border pixels with a specific demosaicing algorithm

### Demosaicing in GigEVisionSDK

In the *GigEVisionSDK* the *ImageProcAPI* provides the programming interface for configuring and executing the demosaicing operations within user applications. The bit depths and image types supported are shown in Table 95. For a detailed description on how to use the demosaicing feature please refer to the *GigEVisionSDK API Help* located in the doc folder of the *GigEVisionSDK* installation directory.

Supported image input	Supported bit depth	
	8 bit per channel	16 bit per channel
Monochrome		
Raw Bayer	✓	✓
Color RGB		

Table 95: Demosaicing - supported bit depth and image type

### Demosaicing in GigEVisionClient

In the *GigEVisionClient* the demosaicing options can be accessed in the *Image Processing Properties* panel under *Color*, shown in Figure 159. If not visible, the panel can be enabled by the menu bar entry *Control* ⇒ *Image Processing Properties*.

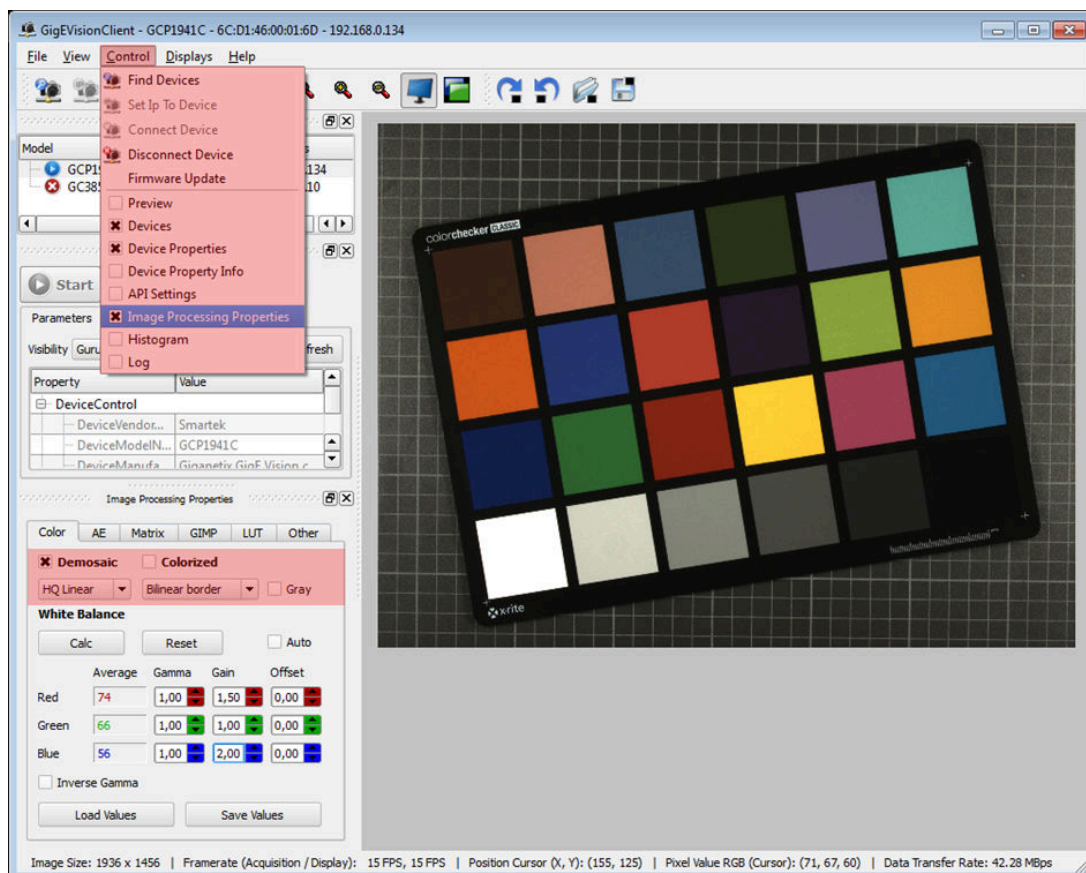


Figure 159: Demosaicing algorithms with border type selection



### 7.2.7 Matrix Multiplication 3x3

N-by-N matrices are commonly used to transform RGB colors, scale them and control hue, saturation and contrast. The *GigEVisionSDK* provides a configurable 3-by-3 matrix for various applications, modifying color images using matrix multiplication operations.



#### Note

For these operations to be correct, they must be operated on linear brightness values. If the input image is in a non-linear brightness space, RGB colors must be transformed into a linear space before these matrix operations are used.

Figure 160 shows how the matrix multiplication is done, where  $m_{xx}$  are the matrix elements,  $R_i / G_i / B_i$  are the input original values for the red, green and blue channel and  $R_o / G_o / B_o$  are the output color values for the red, green and blue channel.

$$\begin{bmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{bmatrix} \times \begin{bmatrix} R_i \\ G_i \\ B_i \end{bmatrix} = \begin{bmatrix} R_o \\ G_o \\ B_o \end{bmatrix}$$

Figure 160: Matrix Multi RGB parameters and results

In effect, this calculates:

$$\begin{aligned} R_o &= m_{00} \cdot R_i + m_{01} \cdot G_i + m_{02} \cdot B_i \\ G_o &= m_{10} \cdot R_i + m_{11} \cdot G_i + m_{12} \cdot B_i \\ B_o &= m_{20} \cdot R_i + m_{21} \cdot G_i + m_{22} \cdot B_i \end{aligned}$$

Common applications for the 3x3 matrix operation are for example color correction, color balancing and the conversion from color to luminance.

### Matrix Multiplication 3x3 in the GigEVisionSDK

In the *GigEVisionSDK* the *ImageProcAPI* provides the programming interface to configure and execute the 3x3 matrix multiplication algorithm. The bit depths and image types supported are shown in Table 96.

Supported image input	Supported bit depth	
	8 bit per channel	16 bit per channel
Monochrome	✓	✓
Raw Bayer	✓	✓
Color RGB	✓	✓

Table 96: Matrix Multiplication - supported bit depth and image type



### Matrix Multiplication 3x3 in the GigEVisionClient

In the *GigEVisionClient* the demosaicing options can be accessed in the *Image Processing Properties* panel under *Matrix*, shown in Figure 161. If not visible, the panel can be enabled by the menu bar entry *Control* ⇒ *Image Processing Properties*.

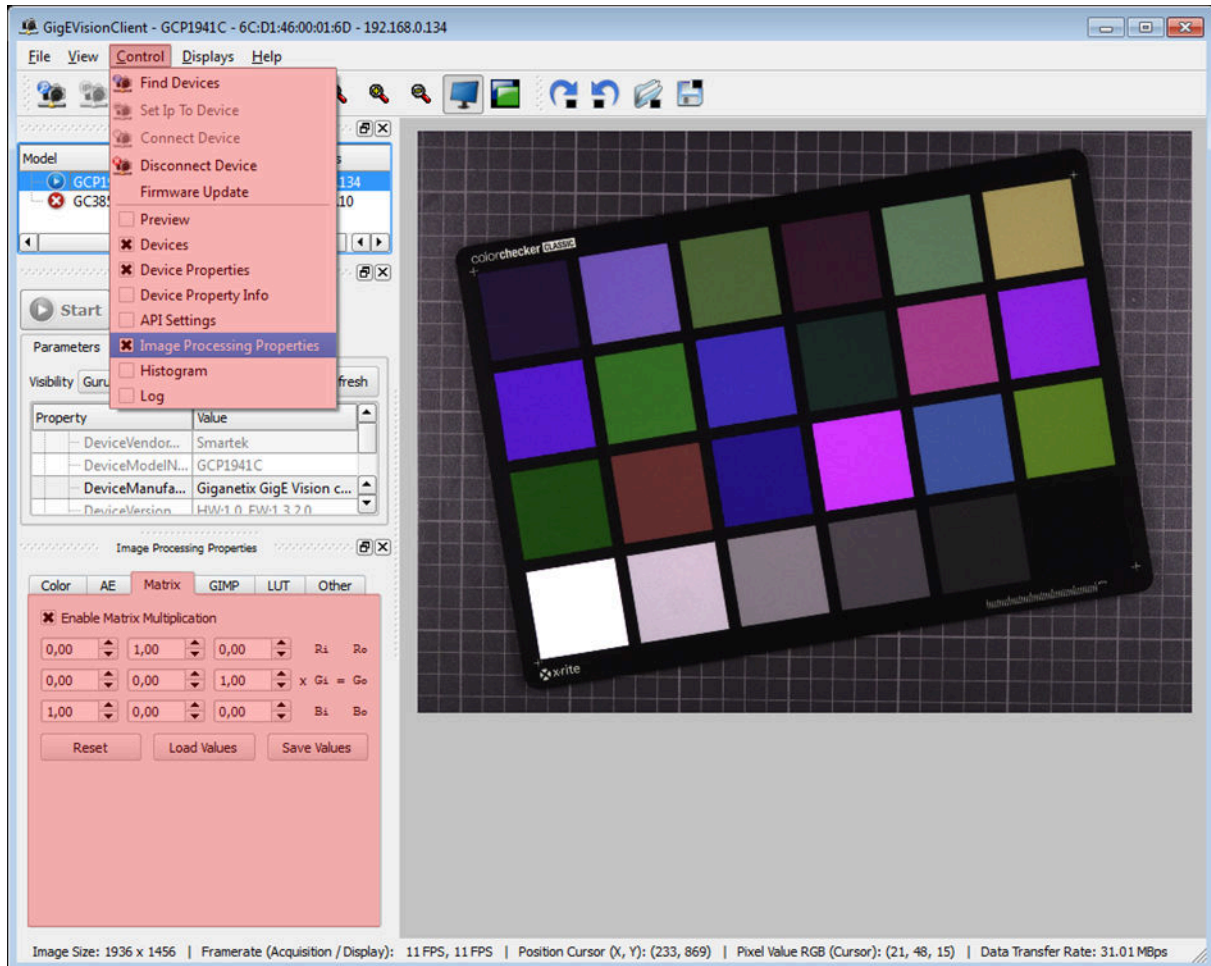


Figure 161: Matrix Multiplication RGB in the GigEVisionClient

- **Enable:** Activate / deactivate the matrix multiplication feature
- **Reset:** Sets matrix coefficients to default values
- **Load Values:** Load a file with user-defined matrix coefficients
- **Save Values:** Save the current matrix coefficients to a file

### 7.2.8 GIMP HSL

The *GIMP HSL* algorithm allows the color manipulation of images based on the HSL color space. The used algorithm is provided by the open source project GIMP and allows the manipulation by the attributes *Hue*, *Saturation* and *Lightness*.

When it comes to manipulating color in images it is often referred to color models or color spaces. Basically a color model describes the way colors can be represented. With understanding of how different color models work, the appropriate color model for specific image processing algorithms can be chosen. The most widely used and best known one is the RGB color model. However, RGB is not always efficient and intuitive in manipulating color.

A more suited color space for manipulating colors is the HSL color space. It was developed to interpret colors in a very similar way as humans do, wherefore color and brightness information are handled separately. The color information is defined by *Hue* and *Saturation*, the brightness information is defined by a *Lightness* value. The HSL color model can be represented by a circle called a color wheel like shown in Figure 162.

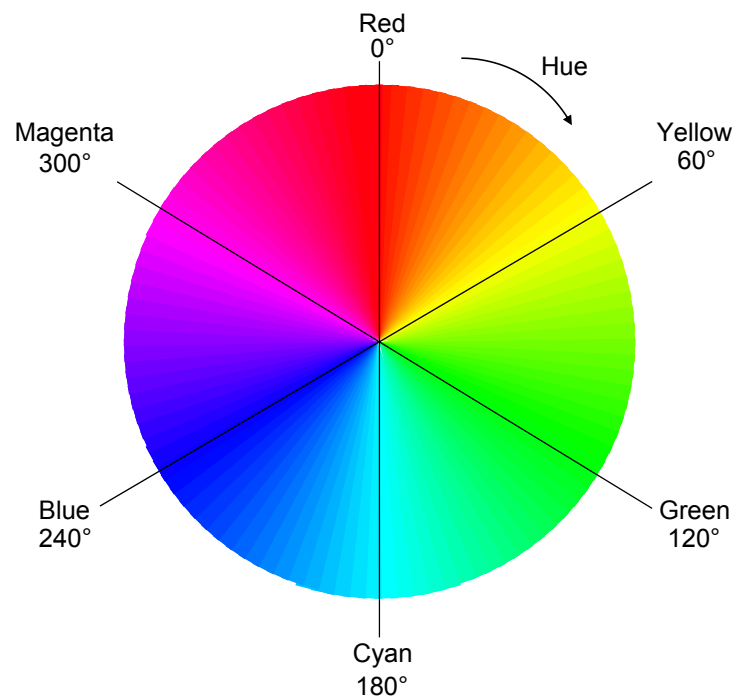


Figure 162: HSL color wheel

*Hue* refers to a specific tone of color: Red, Yellow, Green, Cyan, Blue, Magenta and their blends, the individual colors are arranged in a circle. Their individual position is determined by an angle, ranging from 0° to 360°. Pure red is usually placed at 0°, pure green and pure blue at 120° respectively 240°. Table 97 shows the six base colors.

Hue (angle)	Color
0°	red
60°	yellow
120°	green
180°	cyan
240°	blue
300°	magenta

Table 97: HSL color space

As shown in Figure 162 as well, *Saturation* describes the intensity of a color. It defines how pale or strong a color appears and is the intensity of a Hue from gray. At maximum saturation a color would contain no gray at all, at minimum saturation a color would contain mostly gray. In the HSL color wheel the saturation specifies the distance from the middle of the wheel in percent.

*Lightness* describes how bright or how dark a color appears. It defines how much white or black is contained within a color.

Because of its characteristics of separating color and brightness information, the HSL color space fits for various image processing functions such as convolution, equalization, histograms, which mainly use the brightness information for calculation. As a result, computation performance may also increase due to performing calculation only on one channel.

### Gimp HSL in the GigEVisionSDK

In the *GigEVisionSDK* the *ImageProcAPI* provides the programming interface for configuring and executing *Gimp HSL* algorithm. The bit depth and image type supported are shown in Table 98. For a detailed description on how to use this feature please refer to the *GigEVisionSDK API Help* located in the doc folder of the *GigEVisionSDK* installation directory.

Supported image input	Supported bit depth	
	8 bit per channel	16 bit per channel
Monochrome		
Raw Bayer		
Color RGB	✓	

Table 98: Gimp HSL - supported bit depth and image type

### Gimp HSL in the GigEVisionClient

In the *GigEVisionClient* the *GIMP HSL* manipulation options can be accessed in the *Image Processing Properties* panel under *GIMP*, shown in Figure 163. If not visible, the panel can be enabled by the menu bar entry *Control*  $\Rightarrow$  *Image Processing Properties*. If Master is selected, then values are changed for every channel at once.

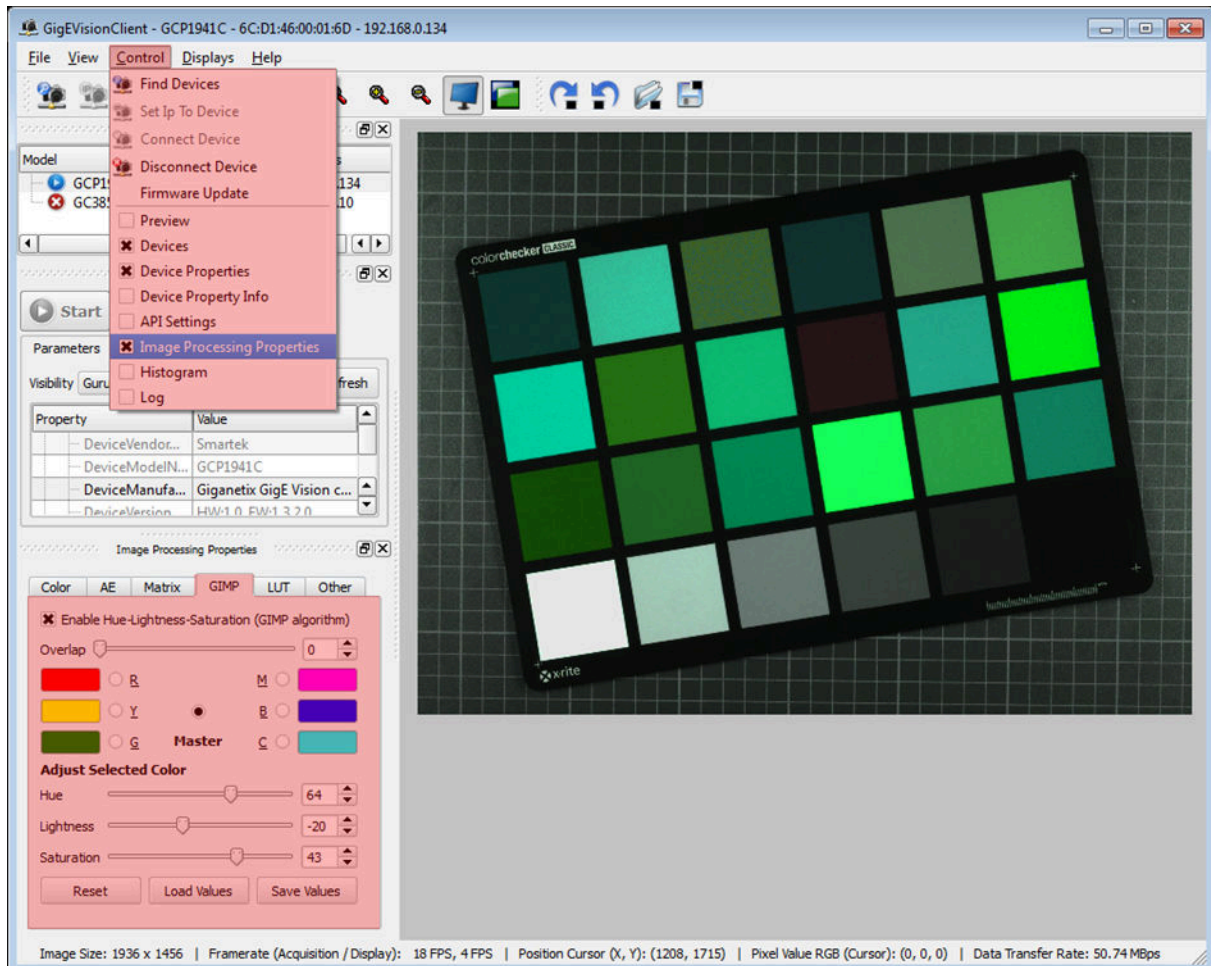


Figure 163: Color GIMP dialog

- **Enable:** activate / deactivate the GIMP Hue / Saturation / Lightness processing
- **Reset:** Sets all settings to default values
- **Load Values:** Load an file with user-defined Hue, Saturation and Lightness values
- **Save Values:** Save the current Hue, Saturation and Lightness values to a file

### 7.2.9 Sharpening

In some situations captured images are blurred, where the reasons may vary; imperfect produced lenses or digital image sensors themselves can blur an image to some degree as well as motion in the scene and image operations which may reduce the sharpness. Especially on Bayer color image sensors, where the missing color information is interpolated, a loss of sharpness is unavoidable.

Sharpening emphasizes edges and fine details in the image, enhancing its visual quality. The image seems sharper, but no new details are actually created.

Figure 164 demonstrates the sharpening algorithm of the *ImageProcAPI*. On the left the original image is displayed, on the right the sharpened image with an applied sharpen factor of 1.0. As result the output image appears sharper in comparison to the original one.

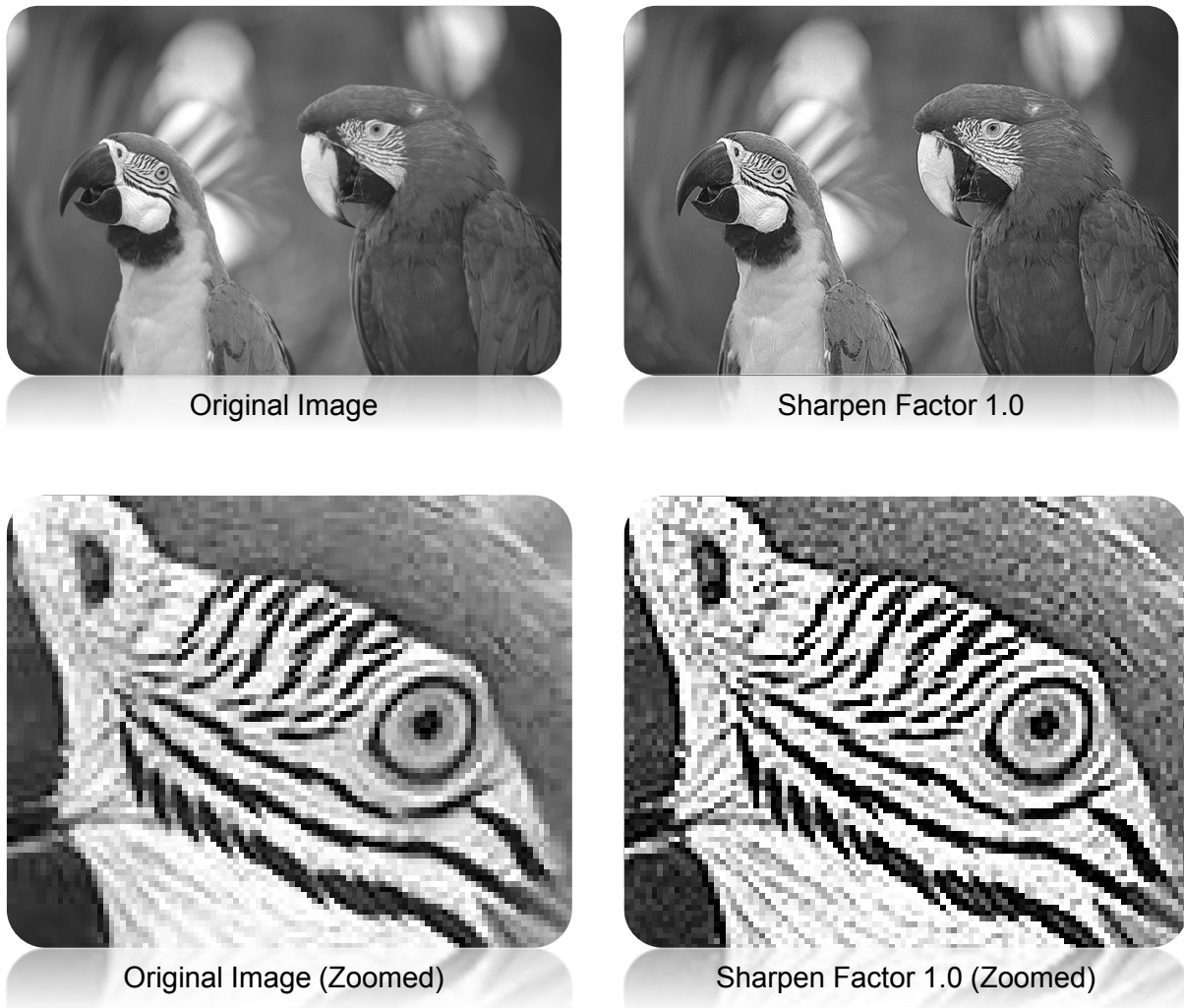


Figure 164: Example of sharpening algorithm (factor 1.0)



### Sharpening in the GigEVisionSDK

In the *GigEVisionSDK* the *ImageProcAPI* provides the programming interface for configuring and executing the sharpening algorithm. The bit depths and image types supported are shown in Table 99. For a detailed description on how to use this feature please refer to the *GigEVisionSDK API Help* located in the doc folder of the *GigEVisionSDK* installation directory.

Supported image input	Supported bit depth	
	8 bit per channel	16 bit per channel
Monochrome	✓	✓
Raw Bayer		
Color RGB	✓	✓

Table 99: Sharpening - supported bit depth and image type

### Sharpening in the GigEVisionClient

In the *GigEVisionClient* the *Image Sharpen* options can be accessed in the *Image Processing Properties* panel under *Other*, shown in Figure 165. If not visible, the panel can be enabled by the menu bar entry *Control* ⇒ *Image Processing Properties*.

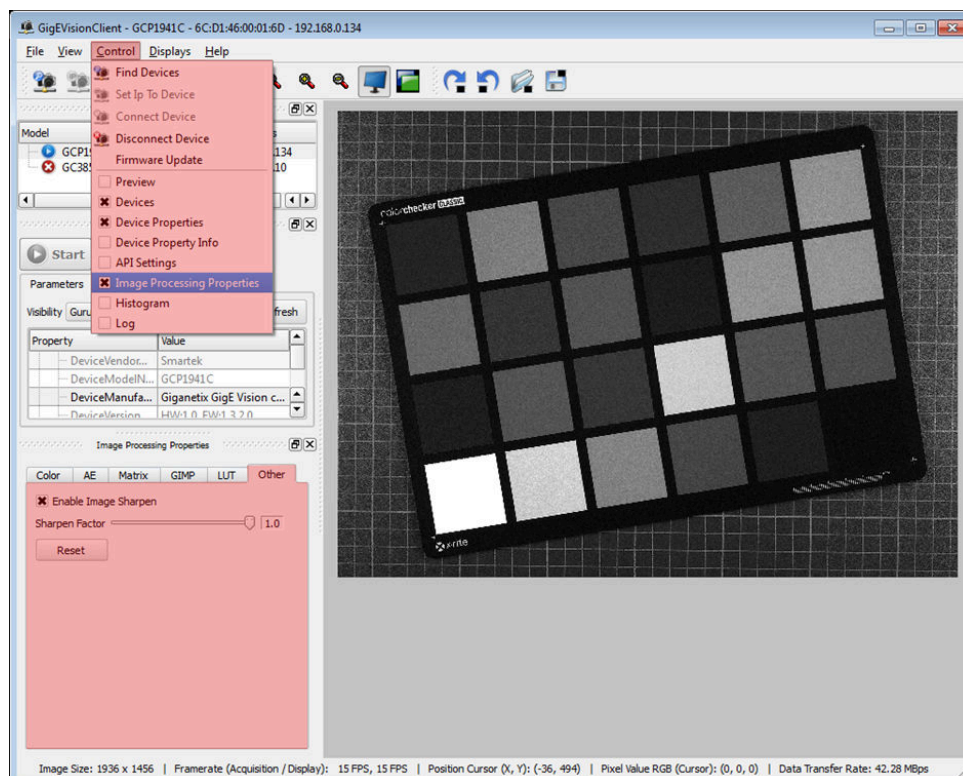


Figure 165: Sharpening in the GigEVisionClient

- **Enable:** Activate / deactivate the image sharpening feature
- **Reset:** Sets the sharpen factor to the default value

### 7.2.10 RGB to Grayscale Conversion

Color images often have to be converted to grayscale, providing input data for subsequent digital image processing like edge detection filters, OCR etc. The RGB-to-grayscale conversion performs a reduction of the RGB color data into a single channel luminance image.

Figure 166 shows an example of RGB-to-gray conversion. The image on the left represents the original RGB color image. The output grayscale image on the right is the result of the conversion process.



Figure 166: Example of RGB-to-gray conversion

### RGB-to-Gray Conversion in the GigEVisionSDK

In the *GigEVisionSDK* the *ImageProcAPI* provides the programming interface for executing the RGB-to-gray conversion. The bit depths and image types supported are shown in Table 100. For a detailed description on how to use this feature please refer to the *GigEVisionSDK API Help* located in the doc folder of the *GigEVisionSDK* installation directory.

Supported image input	Supported bit depth	
	8 bit per channel	16 bit per channel
Monochrome		
Raw Bayer		
Color RGB	✓	✓

Table 100: RGB to Gray conversion - supported bit depths and image types

### RGB-to-Gray Conversion in the GigE VisionClient

In the *GigE VisionClient* the RGB-to-Gray options can be activated in the *Image Processing Properties* panel under *Color*, shown in Figure 167. If not visible, the panel can be enabled by the menu bar entry *Control* ⇒ *Image Processing Properties*.

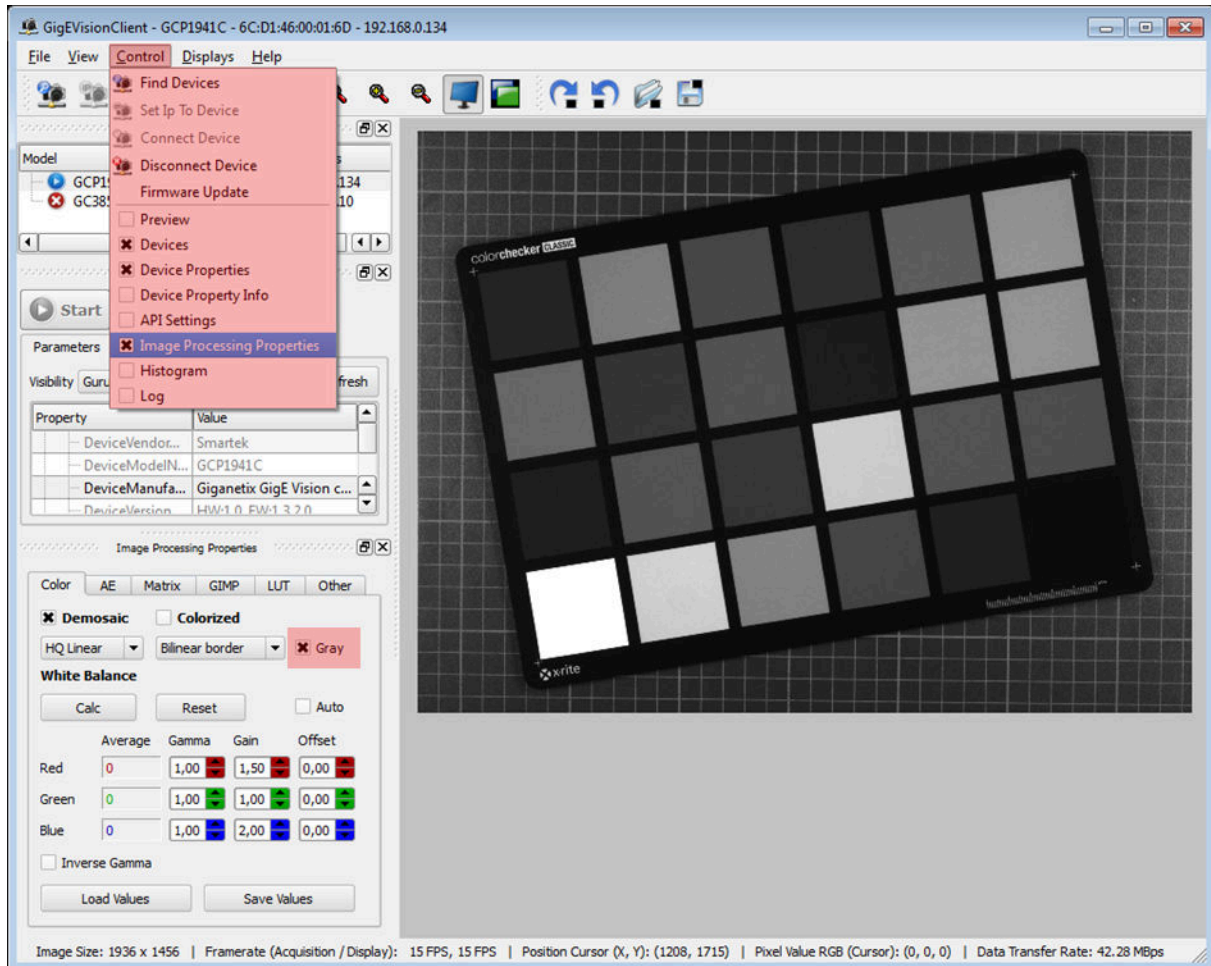


Figure 167: RGB to Gray conversion in GigE VisionClient



### 7.2.11 Bit Depth Conversion

The bit depth of a pixel describes the resolution with which the luminance information is handled. As usual display devices only support 8 bit per channel, the Bit Depth Conversion algorithm there allows the conversion from 16 bit down to 8 bit and vice versa.

Converting images from a higher bit depth to a lower one will lead to reduction of the image size. Please keep in mind that this conversion causes information loss which cannot be recovered by the back conversion to a higher bit depth.

#### Bit Depth Conversion in the GigEVisionSDK

In the *GigEVisionSDK* the *ImageProcAPI* provides the programming interface for converting the bit depth of an image. The bit depths and image types supported are shown in Table 101.

Supported image input	Supported bit depth	
	8 bit per channel	16 bit per channel
Monochrome	✓	✓
Raw Bayer	✓	✓
Color RGB	✓	✓

Table 101: Bit depth conversion - supported bit depth and image type

For a detailed description on how to use this feature please refer to the *GigEVisionSDK API Help* located in the doc folder of the *GigEVisionSDK* installation directory.

#### Bit Depth Conversion in the GigEVisionClient

The Bit Depth Conversion function is automatically applied to 16 bit per channel images to display them on the screen. The inverse conversion from 8 bit to 16 bit is therefore not relevant in the *GigEVisionClient*.

### 7.2.12 Flip / Rotate Transformation

The rotation performs a geometric transform which maps the position  $x_1, y_1$  of a picture element in an input image onto a position  $x_2, y_2$  in an output image by rotating it through a user-specified angle  $\theta$  about an origin  $O$ . When acquiring images from camera, the orientation might not be correct. Rotation / flipping algorithm provides user with possibility to rotate or flip image. Figure 168 demonstrates all flipping cases and Figure 169 demonstrates all rotation cases.

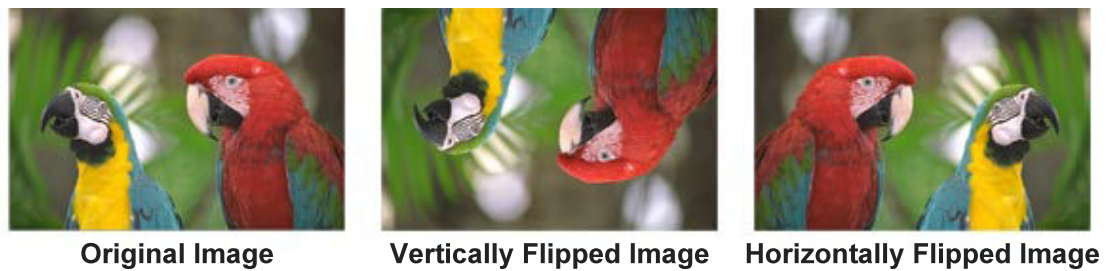


Figure 168: Example of image flipping



Figure 169: Example of image rotation

### Flipping / Rotating in the GigEVisionSDK

In the *GigEVisionSDK* the *ImageProcAPI* provides the programming interface for executing the flipping / rotating transformations. The bit depths and image types supported are shown in Table 102. For a detailed description on how to use this feature please refer to the *GigEVisionSDK API Help* located in the doc folder of the *GigEVisionSDK* installation directory.

Supported image input	Supported bit depth	
	8 bit per channel	16 bit per channel
Monochrome	✓	✓
Raw Bayer	✓	✓
Color RGB	✓	✓

Table 102: Flip - Rotate / supported bit depths and image types

### Flip / Rotate transformations in the GigE VisionClient

In the *GigE VisionClient* the Flip / Rotate options can be activated in the *Image Processing Properties* panel under *Color*, shown in Figure 170. If not visible, the panel can be enabled by the menu bar entry *Control* ⇒ *Image Processing Properties*.

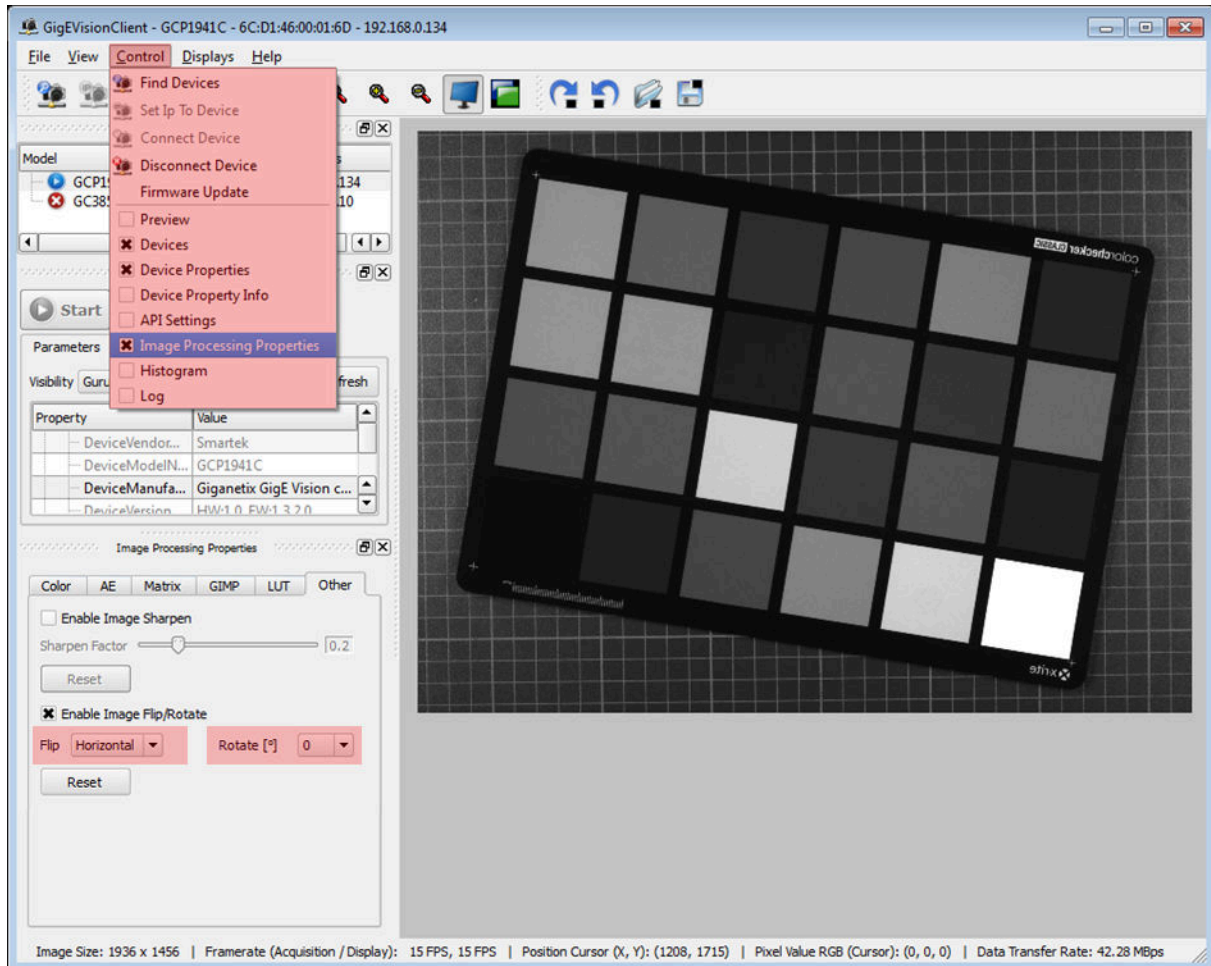


Figure 170: Flip / Rotate transformations in GigE VisionClient

### 7.3 Color Image Processing Pipeline

In the previous chapters the image processing algorithms provided by the *ImageProcAPI* in of the *GigEVisionSDK* have been introduced. Within user applications all image processing algorithms can be combined together in a non-specific order.

The *Image Processing Pipeline* performs the baseline and provides the whole chain in a process optimized way, improving the interaction of all algorithms and thus the overall performance. It takes the raw data produced by a camera sensor and generates the digital image that will then undergo further processing, is viewed by the user and/or stored to a nonvolatile memory.

The preconfigured imaging pipeline supported by the *ImageProcAPI* is illustrated in Figure 171.



Figure 171: Image Processing Pipeline

The order of the algorithms in the color image pipeline provided by the *GigEVisionSDK* is fixed and cannot be modified, only the parameters and the execution of each algorithm can be configured. For other cases a custom image processing pipeline can be combined by the available algorithms in a preferred order.

#### Color Image Processing Pipeline in GigEVisionSDK

In the *GigEVisionSDK* the *ImageProcAPI* provides the programming interface for executing the predefined color image processing pipeline within user applications. The bit depths and image types supported are shown in Table 103. For a detailed description on how to use this feature please refer to the *GigEVisionSDK API Help* located in the doc folder of the *GigEVisionSDK* installation directory.

Supported image input	Supported bit depth	
	8 bit per channel	16 bit per channel
Monochrome	✓	✓
Raw Bayer	✓	✓
Color RGB		

Table 103: Color pipeline - supported bit depth and supported image type

#### Color Image Processing Pipeline in GigEVisionClient

The color image processing pipeline is enabled by default for color cameras. The user only can activate or deactivate a specific algorithm or configure the parameters for each algorithm; the order of the pipeline cannot be changed.

## **8 Contact Information**

Published by:

Smartek d.o.o.  
Dobrise Cesarica 5  
HR-40000 Cakovec  
Croatia

[www.SMARTEK.vision](http://www.SMARTEK.vision)

Email:     [info@SMARTEKvision.com](mailto:info@SMARTEKvision.com)  
Tel:        +49 (89) 381 53 30 - 57  
Fax:        +49 (89) 381 53 30 - 58

Copyright © 2015 by Smartek d.o.o. All rights reserved.  
For further information please contact our sales partners.

## 9 Revision History

Version number	Date	Changes
2.1.1	2014-09-15	Preliminary version of the document.
2.1.2	2014-10-02	Added chapter "Image Processing on Camera".
2.1.3	2014-10-17	Added subchapter "Center X / Y".
2.1.4	2015-02-13	Added subchapter "Frame Transfer Delay"; Mono10/12Packed support; updated camera features list.
2.2.0	2015-03-18	Replaced subchapters "Area of Interest Control (AOI)" with "Region of Interest" and "Center X / Y" with "ROI Centering"; added subchapter "Multiple ROI"; added subchapter "Automatic Exposure and Gain Control"; updated camera features list.
2.3.0	2015-07-13	Updated chapter "Sensor Information and Technical Specification" with GC1932 camera model; updated chapter "List of Supported Features".
2.4.0	2015-12-01	Updated chapter "Sensor Information and Technical Specification" with GCP2061 and GCP2461 camera models; updated references to RoHS certification; updated chapter "List of Supported Features".