smartek vision

User Manual

twentynine Camera Family

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

Contents

1 Platform Specification

The SMARTEK Vision twentynine camera family offers an affordable and easy to use set of digital cameras designed to meet demanding high quality machine vision applications, conforming to the industrial GigE Vision and USB3 Vision standard. The compact housing fits almost every space critical application. Focusing on a wide selection of SONY and ON Semiconductor CMOS sensors, the twentynine platform delivers images with high sensitivity and low noise on a broad range of image resolutions at high framerates. Excellent price to performance ratio makes this portfolio the perfect choice for every demanding user.

SMARTEK Vision twentynine cameras combine standard Gigabit Ethernet and USB3.0 technology with the CameraSuite image acquisition software to reliably capture and transfer images from the camera to the PC. All twentynine cameras are supported by one Software Development Kit as well as a large number of 3rd-party libraries compliant to the GigE Vision and USB3 Vision standards. To use our devices with other software than provided by SMARTEK Vision, please check the documentation of your 3rd-party provider.

Key benefits & features:

- Powerful hardware in miniature 29x29 mm footprint
- Latest CMOS Global Shutter sensors with SONY Pregius series
- Resolutions from 2.3MP up to 5MP on standard C-Mount
- Fully GigE Vision and USB3 Vision compliant:
  - High data rates and plug & play
  - Long cable length up to 100 m (GigE Vision)
- Opto-isolated input and output
- Standard C-Mount lens adapter
- Low power consumption
- Enhanced set of supported features:
  - Automatic Exposure / Gain Control
  - Multiple Area of Interest Support
  - Gamma Control
  - 10-/12-Bit Lookup-Table
  - Chunk Data Control
- Affordable cabling and industrial connectors:
  - EIAJ (Hirose) 6 pin and screw mount RJ45 / USB3.0 Micro-B
- Comprehensive SDK for Windows and Linux, supporting C, C++, .NET and Delphi
- Flexible customization concept
- 3 years warranty

1.1 twentynine Standard

The twentynine camera series with standard housing represents the regular camera design for the xCC series, with the main focus on a small form factor, offering the comprehensive camera electronics in a small 29x29x43 mm footprint. Table 1 contains an overview about the model specific specifications.

1.1.1 List of Specifications

	GCC	UCC
Data Interface	**GigE Vision**	**USB3 Vision**
External dimensions (H x W x L)	29 x 29 x 43 [mm]	
Housing	Black anodized aluminum case	
Weight	Approx. 90g	
Storage temperature	-30°C to +60°C	
Operating housing temperature	0°C to +50°C	
Operating humidity	20% to 80%, relative, non-condensing	
Storage humidity	20% to 80%, relative, non-condensing	
Power requirement	10V to 24V DC or Power over Ethernet (PoE)	Via USB3.0 interface
Lens mount	C-Mount	
Connectors	Screw mount Ethernet RJ45 (Communication and Data), Circular Hirose 6 pin (Power and I/O-Interface)	
Digital input	1 input channel, opto-isolated	
Digital output	1 output channel, opto-isolated	
Data/Control Interface	1GBase-T / 100Base-T (RJ45)	USB3.0 (Micro-B)
Conformity1	CE, RoHS II, GenICam	
Conformity2	GigE Vision, PoE (IEEE802.3af)	USB3 Vision

Table 1: Mechanical and electrical specifications

1.1.2 Technical Drawings

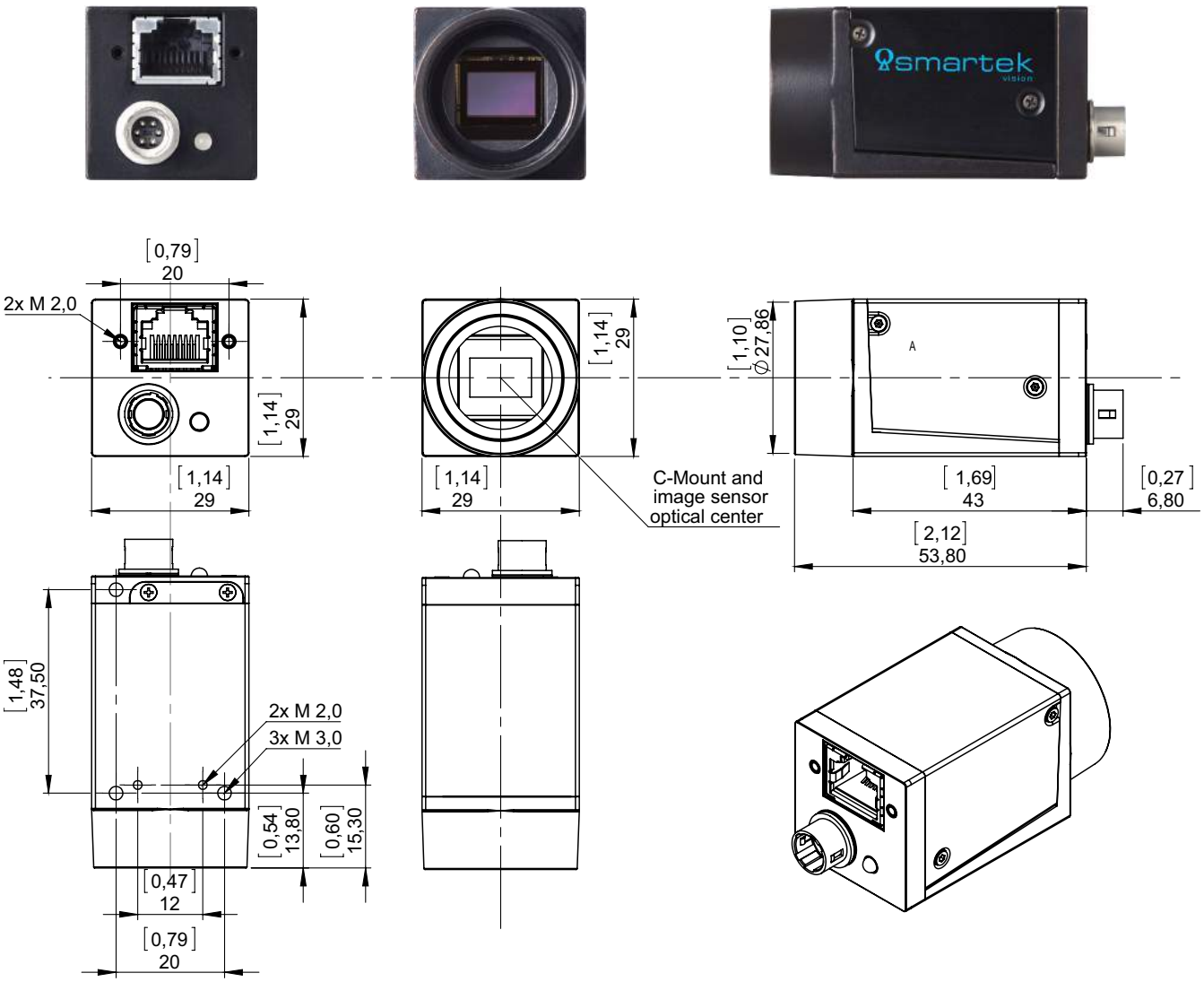


Figure 1: Technical measures of the GCC camera housing (all dimensions are in mm [inch])

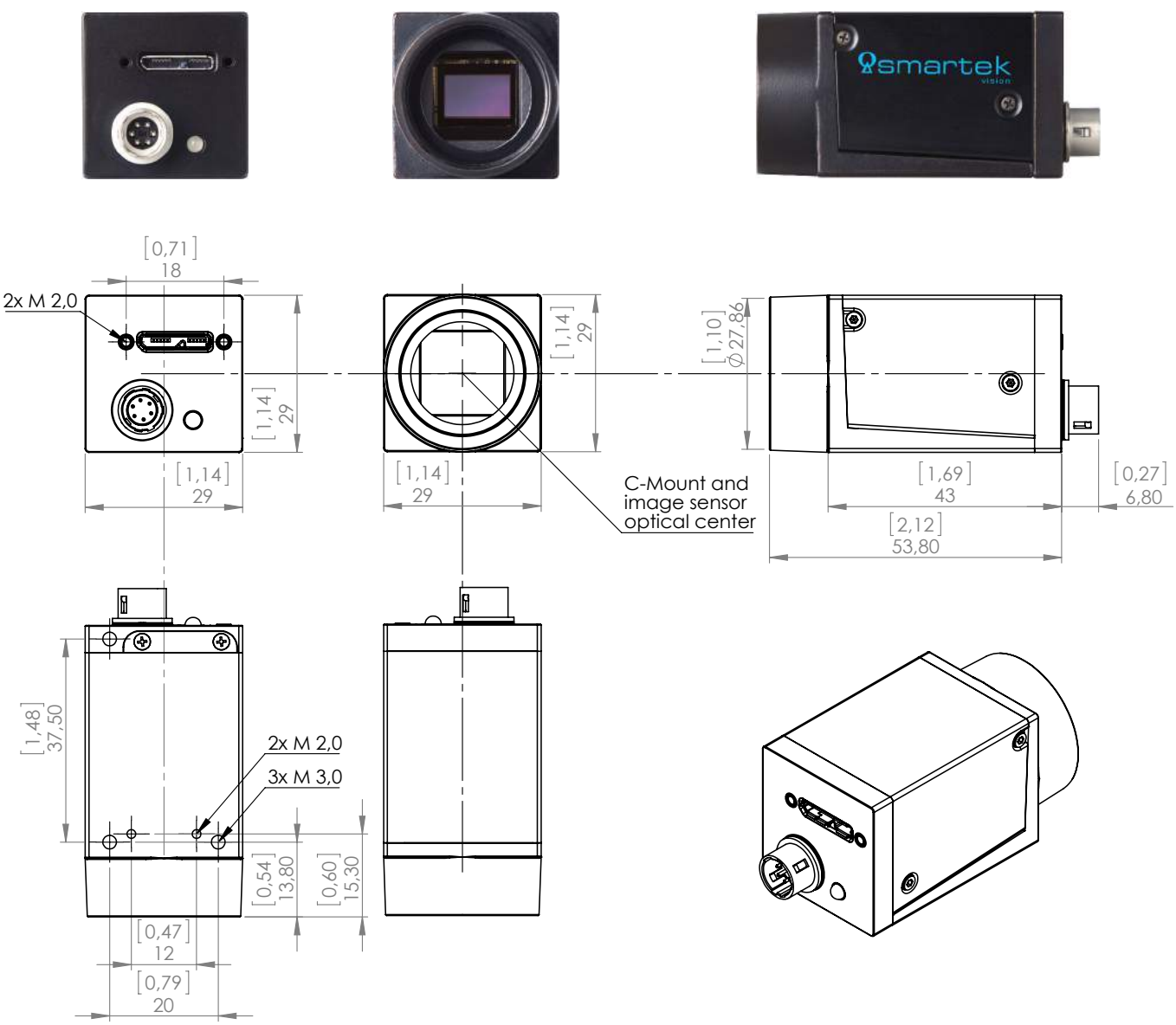


Figure 2: Technical measures of the UCC camera housing (all dimensions are in mm [inch])

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 administrated 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.
- Uses GenICam™ generic programming interface

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 USB3 Vision

USB3 Vision is a communication interface standard for high-bandwidth industrial cameras based on the USB3.0 interface. Similar to the GigE Vision standard it unifies different protocols used in machine vision industrial applications and makes hardware and software from various vendors interoperate seamlessly over a USB3.0 connection. USB3 Vision is administrated by the *Automated Imaging Association* (AIA).

Features of the USB3 Vision standard:

- Very-high data transfer rates - up to 350 MB/s
- Plug-and-play interface for easy use
- Uses GenICam™ generic programming interface
- Based on established standard USB3.0

1.2.3 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.

1.2.4 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 Supported Third-Party Software

The twentynine 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

1.4 IR-Cut Filter Specification

The spectral sensitivity of the 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 3 below shows the transmission curve of the filter used in the twentynine camera family.

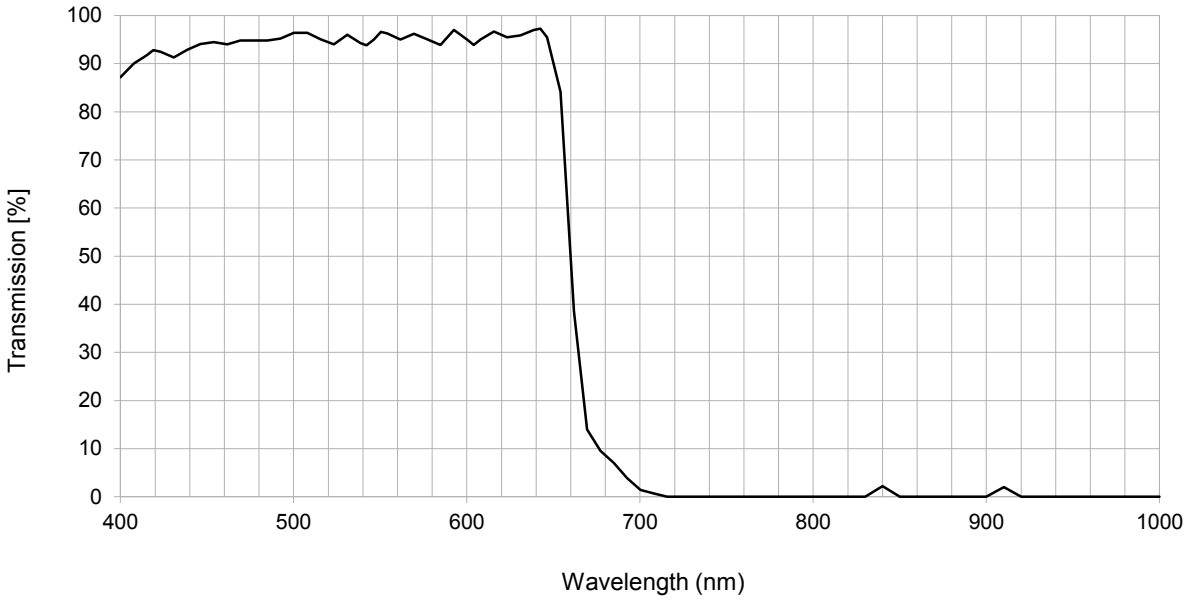


Figure 3: IR-cut filter specification

1.5 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.

1.6 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.

1.7 Temperature 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, SMARTEK 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 4 gives an example of the thermal behavior of a twentynine camera mounted to different heat conductors, described in Table 3. The used camera is a GCC1932M with IMX249 CMOS Imaging sensor, which was chosen as one with typical power consumption (3.2 Watts) in the twentynine lineup.

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

Table 3: Description of the curves in Figure 4

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 significant; the camera which is not mounted to any heat conductor shows after one hour in an environment with 22.5°C a device temperature of 50.4°C. With a small aluminum heat conductor the camera temperature drops about 12°C to 15°C, depending on the connection to the heat sink.

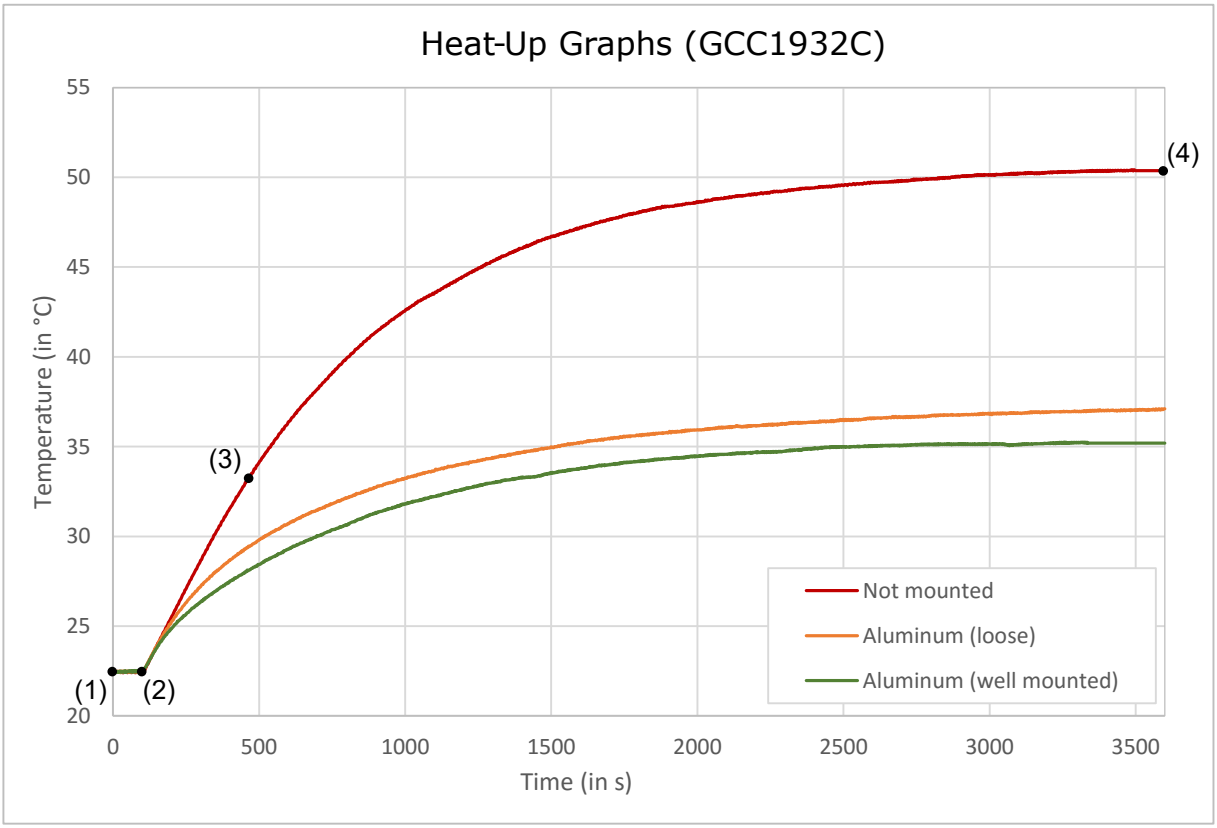


Figure 4: Example of heating behavior of a camera with different thermal connections

1.8 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.

1.9 Declarations of Conformity

1.9.1 CE

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

Product: GCC Digital Gigabit Ethernet Camera
UCC Digital USB Camera

Type Family: twentynine GigE
twentynine USB

Type of Equipment: GCC1931C, GCC1931M, GCC1932C, GCC1932M, GCC2061C, GCC2061M, GCC2062C, GCC2062M, GCC2461C, GCC2461M, GCC2462C, GCC2462M, UCC1931C, UCC1931M UCC1932C, UCC1932M, UCC2061C, UCC2061M, UCC2062C, UCC2062M, UCC2461C, UCC2461M, UCC2462C, UCC2462M.

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

Reference No.	**Title:**
2014/30/EU	Electromagnetic Compatibility (EMC directive)

Following standards or normative documents:
EN 61000-6-3:2007 + A1:2011
EN 61000-6-2:2005

Certificates / Test reports:
C251-0072/17 / T251-0272/17
C251-0071/17 / T251-0275/17

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
CEO
Smartek d.o.o.

1.9.2 RoHS II

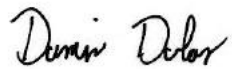
Manufacturer: Smartek d.o.o
Dobrise Cesarica 5
HR-40000 Cakovec
Croatia

Product: GCC Digital Gigabit Ethernet Camera
UCC Digital USB Camera

Type Family: twentynine GigE
twentynine USB

Type of Equipment: GCC1931C, GCC1931M, GCC1932C, GCC1932M, GCC2061C, GCC2061M, GCC2062C, GCC2062M, GCC2461C, GCC2461M, GCC2462C, GCC2462M, UCC1931C, UCC1931M UCC1932C, UCC1932M, UCC2061C, UCC2061M, UCC2062C, UCC2062M, UCC2461C, UCC2461M, UCC2462C, UCC2462M.

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



Damir Dolar
CEO
Smartek d.o.o.

2 Model Overview

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

2.1 GCC1931 / UCC1931

	GCC	UCC
Image Sensor	Sony IMX174	
Chromatics	Monochrome, Color	
Sensor type	CMOS	
Sensor resolution (W x H)	1936 x 1216	
Optical size	1/1.2"	
Pixel size (in µm)	5.86 x 5.86	
Analog gain (in dB)	0 to 24	
Shutter	Global Shutter	
Exposure time	26µs to 10s	19µs to 10s
Max. frame rate (in Hz)		
8Bit	52	164
ADC bit depth	12bit	10bit
Pixel data formats (mono model)	Mono8, Mono12Packed	Mono8, Mono10Packed
Pixel data formats (color model)	Bayer8, Bayer12Packed	Bayer8, Bayer10Packed
Synchronization	Free run, external and software trigger (single shot, multi shot)	
Exposure control	Freely programmable via GigE/USB3 Vision interface	
Power consumption (AUX @12V)	3.2W	—
Power consumption (PoE / USB)	3.8W	2.9W

Table 4: Model specific specification of GCC1931 / UCC1931

Relative Response

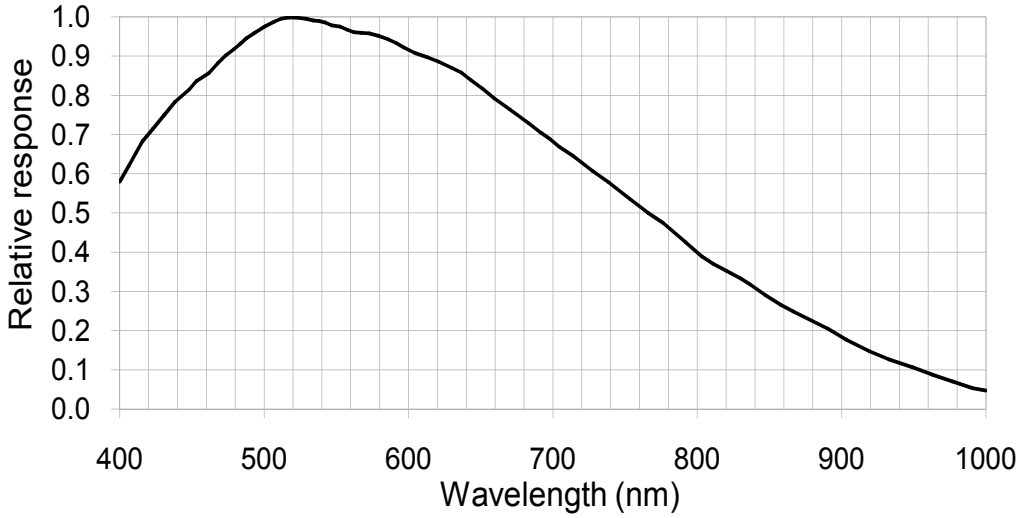


Figure 5: Relative response of GCC1931 / UCC1931 Monochrome (from sensor datasheet)

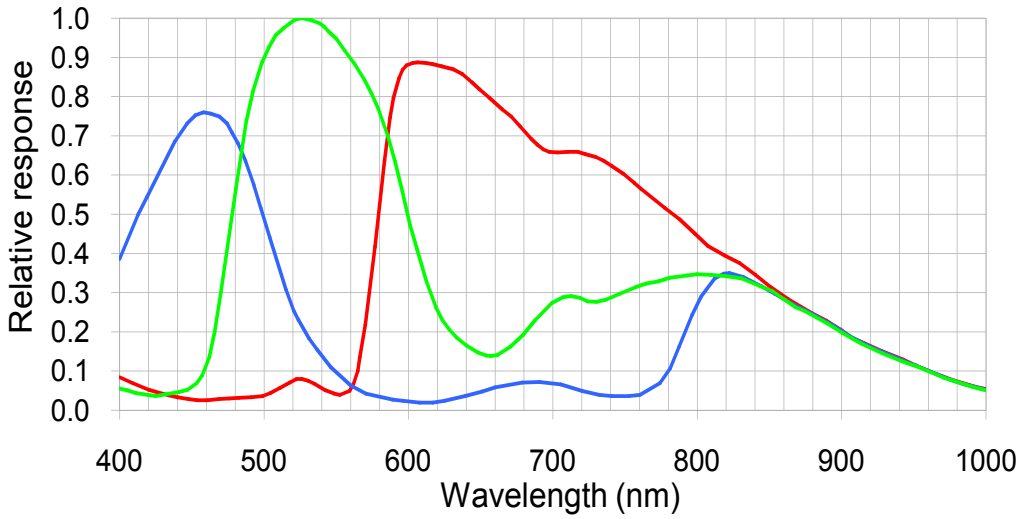


Figure 6: Relative response of GCC1931 / UCC1931 Color (from sensor datasheet)

2.2 GCC1932 / UCC1932

	GCC	UCC
Image Sensor	Sony IMX249	
Chromatics	Monochrome, Color	
Sensor type	CMOS	
Sensor resolution (W x H)	1936 x 1216	
Optical size	1/1.2"	
Pixel size (in µm)	5.86 x 5.86	
Analog gain (in dB)	0 to 24	
Shutter	Global Shutter	
Exposure time	33µs to 10s	33µs to 10s
Max. frame rate (in Hz)		
8Bit	41	41
ADC bit depth	10bit	10bit
Pixel data formats (mono model)	Mono8, Mono12Packed	Mono8, Mono10Packed
Pixel data formats (color model)	Bayer8, Bayer12Packed	Bayer8, Bayer10Packed
Synchronization	Free run, external and software trigger (single shot, multi shot)	
Exposure control	Freely programmable via GigE/USB3 Vision interface	
Power consumption (AUX @12V)	3.1W	—
Power consumption (PoE / USB)	3.7W	2.3W

Table 5: Model specific specification of GCC1932 / UCC1932

Relative Response

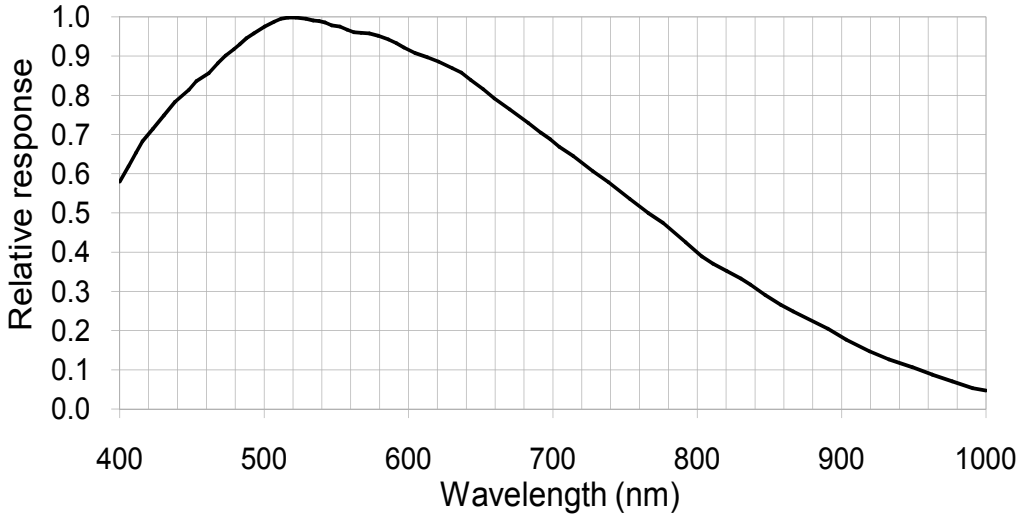


Figure 7: Relative response of GCC1932 / UCC1932 Monochrome (from sensor datasheet)

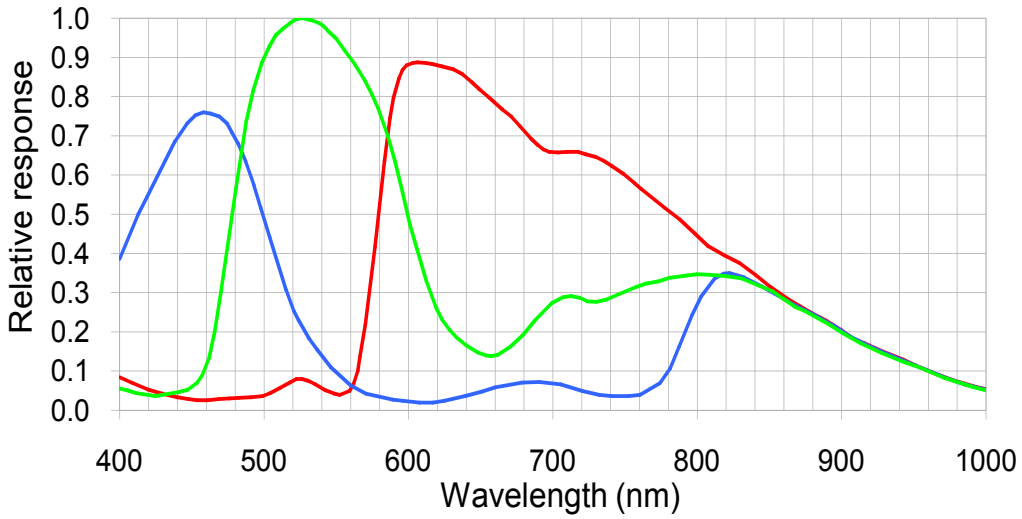


Figure 8: Relative response of GCC1932 / UCC1932 Color (from sensor datasheet)

2.3 GCC2061 / UCC2061

	GCC	UCC
Image Sensor	Sony IMX252	
Chromatics	Monochrome, Color	
Sensor type	CMOS	
Sensor resolution (W x H)	2064 x 1544	
Optical size	1/1.8"	
Pixel size (in µm)	3.45 x 3.45	
Analog gain (in dB)	0 to 24	
Shutter	Global Shutter	
Exposure time	25µs to 10s	19µs to 10s
Max. frame rate (in Hz)		
8Bit	38	120
ADC bit depth	12bit	10bit
Pixel data formats (mono model)	Mono8, Mono12Packed	Mono8, Mono10Packed
Pixel data formats (color model)	Bayer8, Bayer12Packed	Bayer8, Bayer10Packed
Synchronization	Free run, external and software trigger (single shot, multi shot)	
Exposure control	Freely programmable via GigE/USB3 Vision interface	
Power consumption (AUX @12V)	3.4W	—
Power consumption (PoE / USB)	4.1W	3.1W

Table 6: Model specific specification of GCC2061 / UCC2061

Relative Response

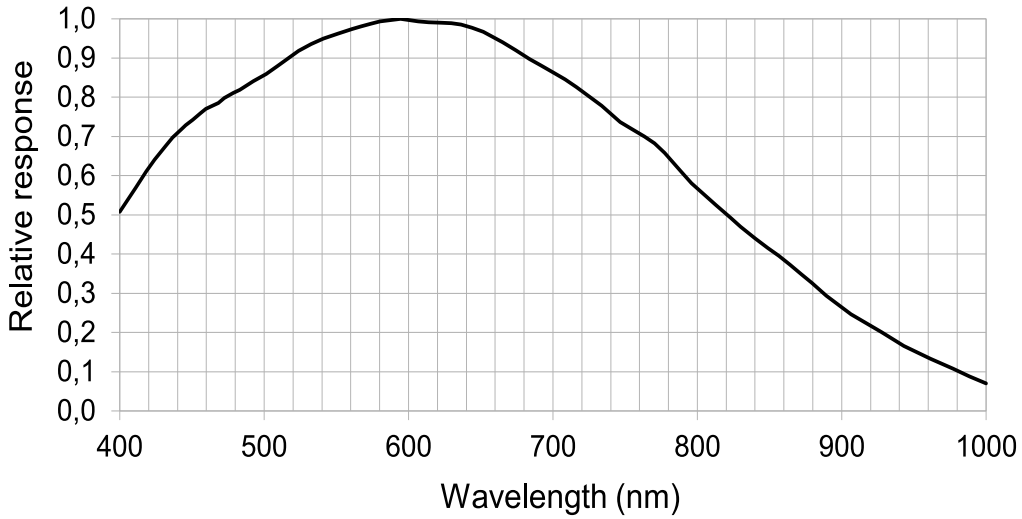


Figure 9: Relative response of GCC2061 / UCC2061 Monochrome (from sensor datasheet)

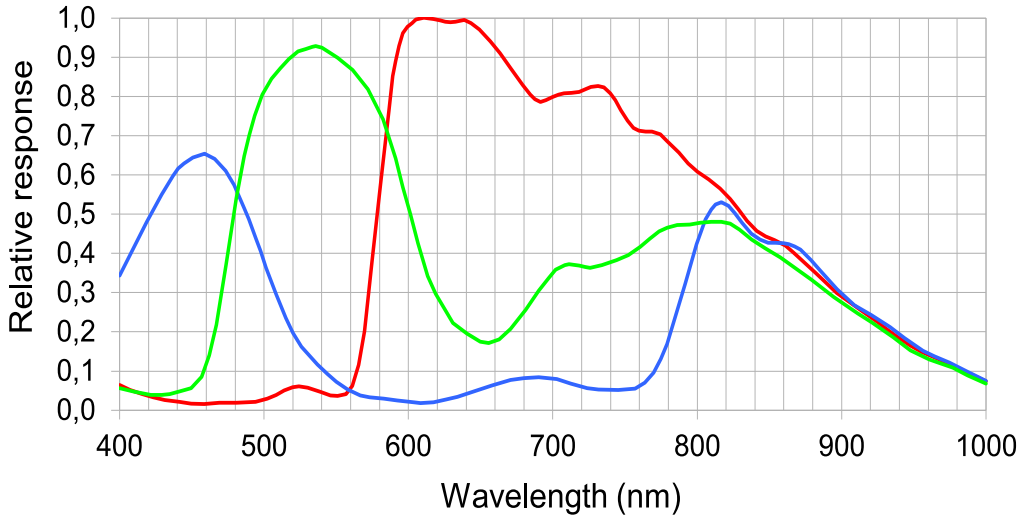


Figure 10: Relative response of GCC2061 / UCC2061 Color (from sensor datasheet)

2.4 GCC2062 / UCC2062

	GCC	UCC
Image Sensor	Sony IMX265	
Chromatics	Monochrome, Color	
Sensor type	CMOS	
Sensor resolution (W x H)	2064 x 1544	
Optical size	1/1.8"	
Pixel size (in µm)	3.45 x 3.45	
Analog gain (in dB)	0 to 24	
Shutter	Global Shutter	
Exposure time	25µs to 10s	25µs to 10s
Max. frame rate (in Hz)		
8Bit	38	55
ADC bit depth	12bit	12bit
Pixel data formats (mono model)	Mono8, Mono12Packed	Mono8, Mono10Packed
Pixel data formats (color model)	Bayer8, Bayer12Packed	Bayer8, Bayer10Packed
Synchronization	Free run, external and software trigger (single shot, multi shot)	
Exposure control	Freely programmable via GigE/USB3 Vision interface	
Power consumption (AUX @12V)	3.4W	—
Power consumption (PoE / USB)	4.0W	2.8W

Table 7: Model specific specification of GCC2062 / UCC2062

Relative Response

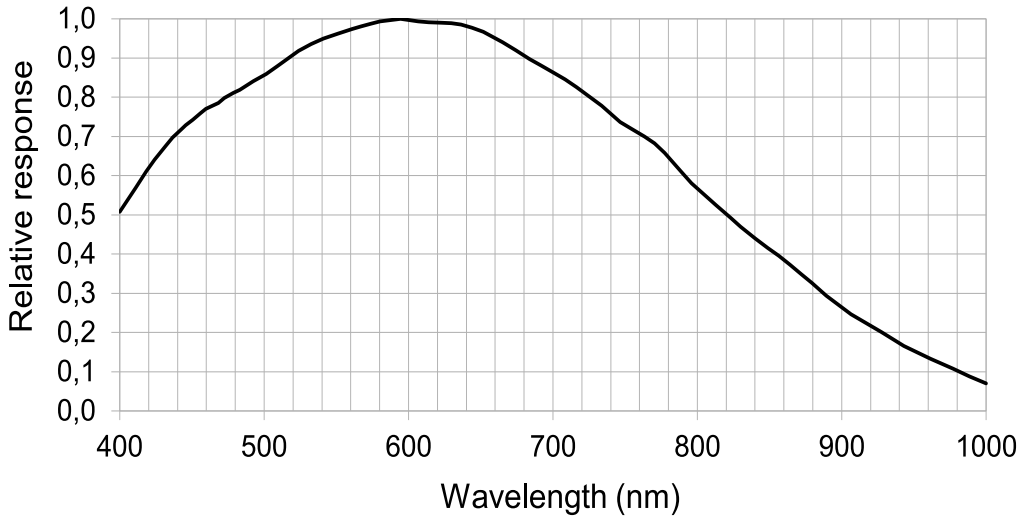


Figure 11: Relative response of GCC2062 / UCC2062 Monochrome (from sensor datasheet)

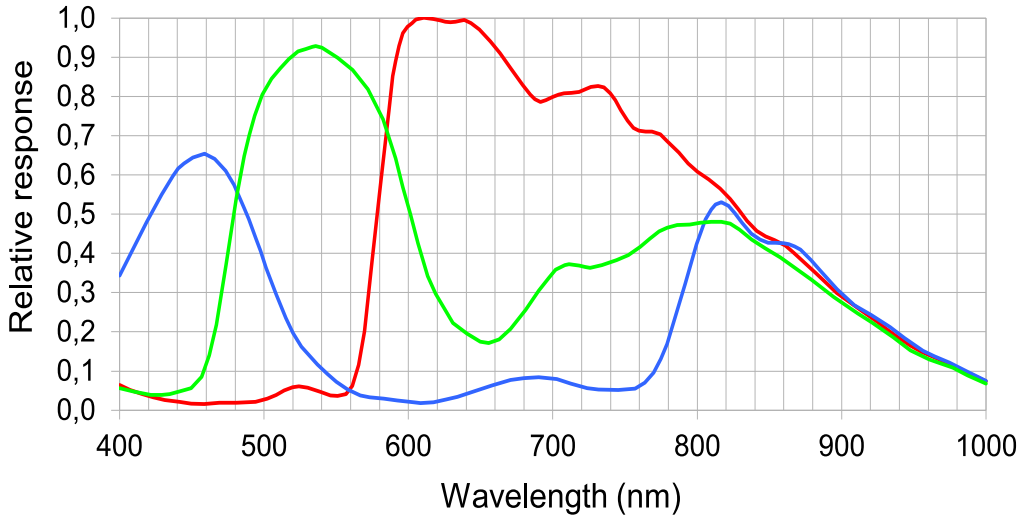


Figure 12: Relative response of GCC2062 / UCC2062 Color (from sensor datasheet)

2.5 GCC2461 / UCC2461

	GCC	UCC
Image Sensor	Sony IMX250	
Chromatics	Monochrome, Color	
Sensor type	CMOS	
Sensor resolution (W x H)	2464 x 2056	
Optical size	2/3"	
Pixel size (in µm)	3.45 x 3.45	
Analog gain (in dB)	0 to 24	
Shutter	Global Shutter	
Exposure time	27µs to 10s	20µs to 10s
Max. frame rate (in Hz)		
8Bit	24	75
ADC bit depth	12bit	10bit
Pixel data formats (mono model)	Mono8, Mono12Packed	Mono8, Mono10Packed
Pixel data formats (color model)	Bayer8, Bayer12Packed	Bayer8, Bayer10Packed
Synchronization	Free run, external and software trigger (single shot, multi shot)	
Exposure control	Freely programmable via GigE/USB3 Vision interface	
Power consumption (AUX @12V)	3.5W	—
Power consumption (PoE / USB)	4.2W	3.2W

Table 8: Model specific specification of GCC2461 / UCC2461

Relative Response

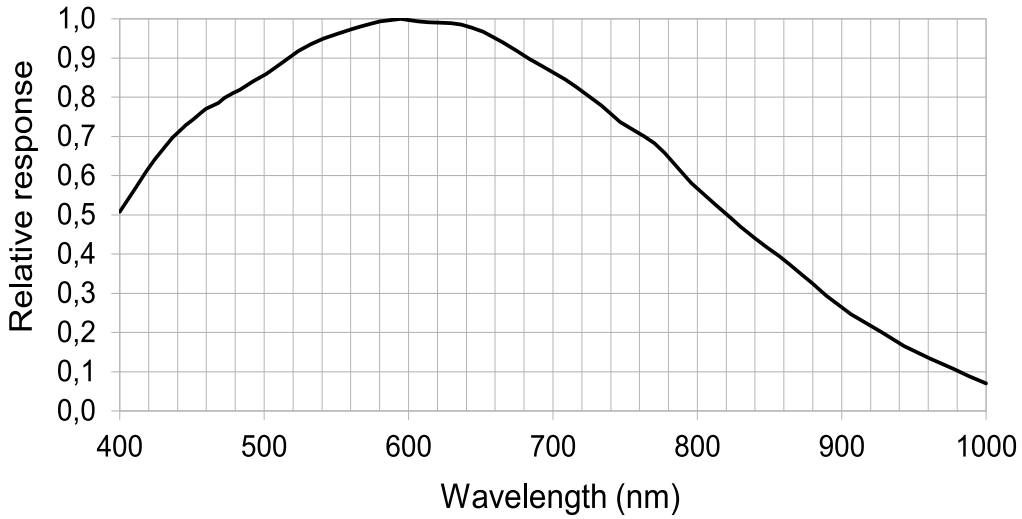


Figure 13: Relative response of GCC2461 / UCC2461 Monochrome (from sensor datasheet)

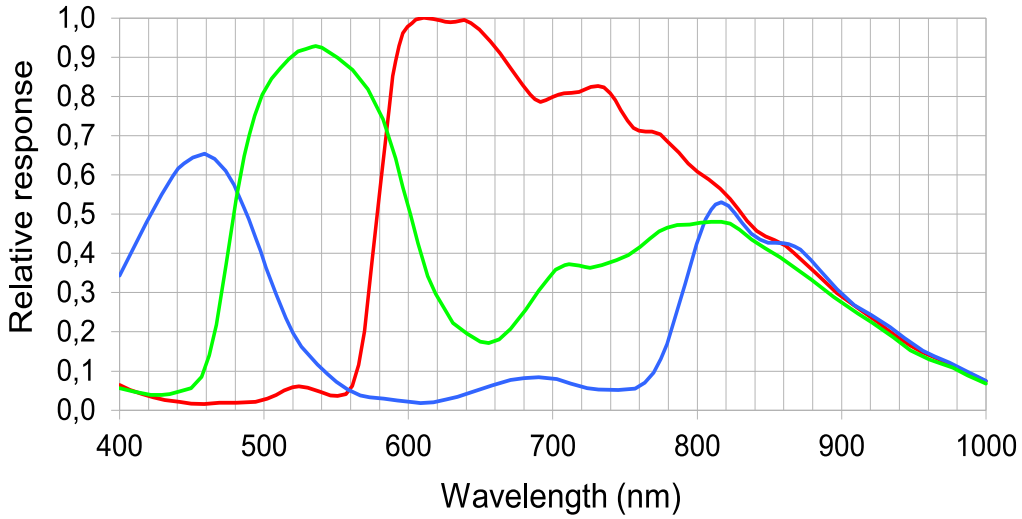


Figure 14: Relative response of GCC2461 / UCC2461 Color (from sensor datasheet)

2.6 GCC2462 / UCC2462

	GCC	UCC
Image Sensor	Sony IMX264	
Chromatics	Monochrome, Color	
Sensor type	CMOS	
Sensor resolution (W x H)	2464 x 2056	
Optical size	2/3"	
Pixel size (in µm)	3.45 x 3.45	
Analog gain (in dB)	0 to 24	
Shutter	Global Shutter	
Exposure time	27µs to 10s	27µs to 10s
Max. frame rate (in Hz)		
8Bit	24	35
ADC bit depth	12bit	12bit
Pixel data formats (mono model)	Mono8, Mono12Packed	Mono8, Mono10Packed
Pixel data formats (color model)	Bayer8, Bayer12Packed	Bayer8, Bayer10Packed
Synchronization	Free run, external and software trigger (single shot, multi shot)	
Exposure control	Freely programmable via GigE/USB3 Vision interface	
Power consumption (AUX @12V)	3.5W	—
Power consumption (PoE / USB)	4.0W	2.8W

Table 9: Model specific specification of GCC2462 / UCC2462

Relative Response

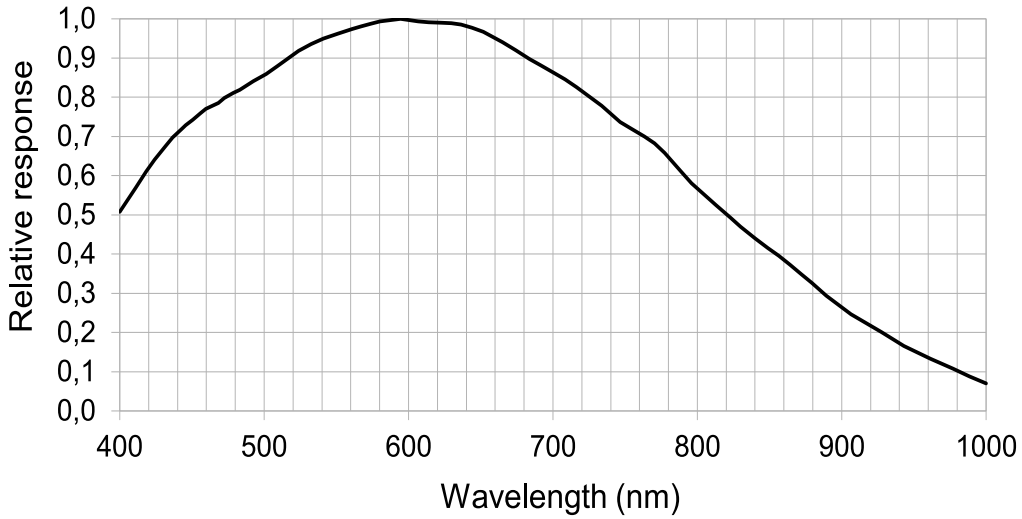


Figure 15: Relative response of GCC2462 / UCC2462 Monochrome (from sensor datasheet)

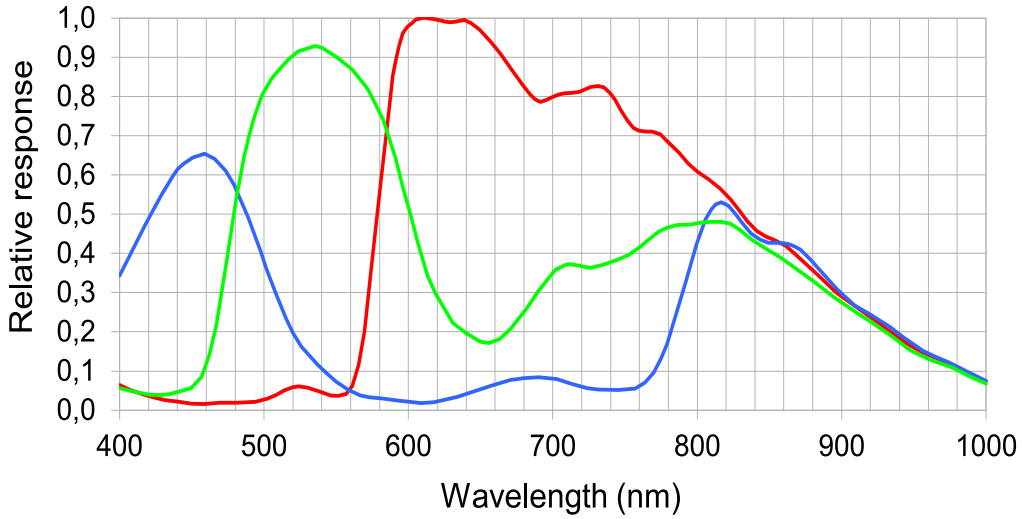


Figure 16: Relative response of GCC2462 / UCC2462 Color (from sensor datasheet)

3 Physical Interfaces

Each camera is equipped with two physical interfaces - a circular Hirose jack providing the camera's power and digital IO lines and a data interface for configuration and streaming. Figure 17 and Figure 18 show the general connecting schemes for USB3 Vision and GigE Vision cameras.

Device	Vision Standard	Interface
twentynine GCC	GigE Vision	Gigabit Ethernet
twentynine UCC	USB3 Vision	USB3.0

Table 10: Data Interfaces of twentynine Series

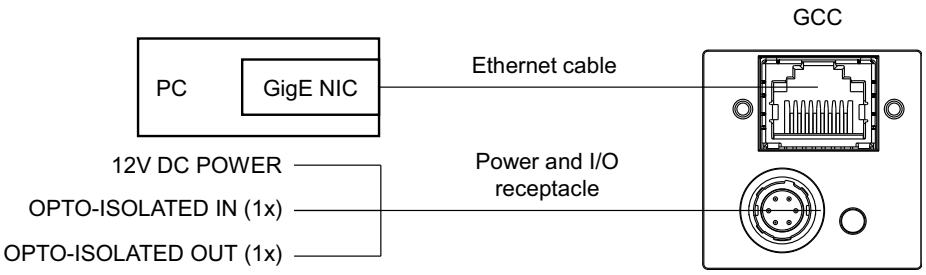


Figure 17: GCC connecting scheme

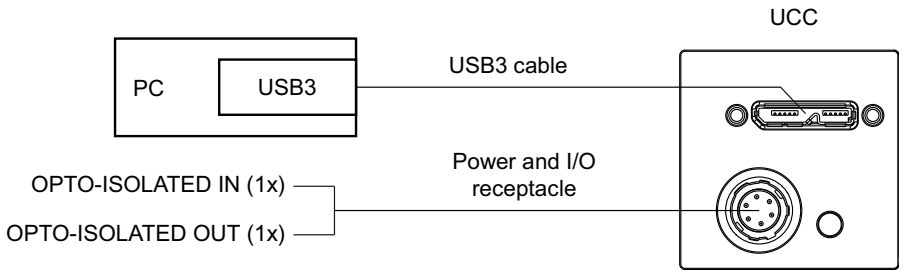


Figure 18: UCC connecting scheme

3.1 Gigabit Ethernet Interface (GCC)

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 11 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 11: Ethernet connector type and assignment

3.1.1 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 19.

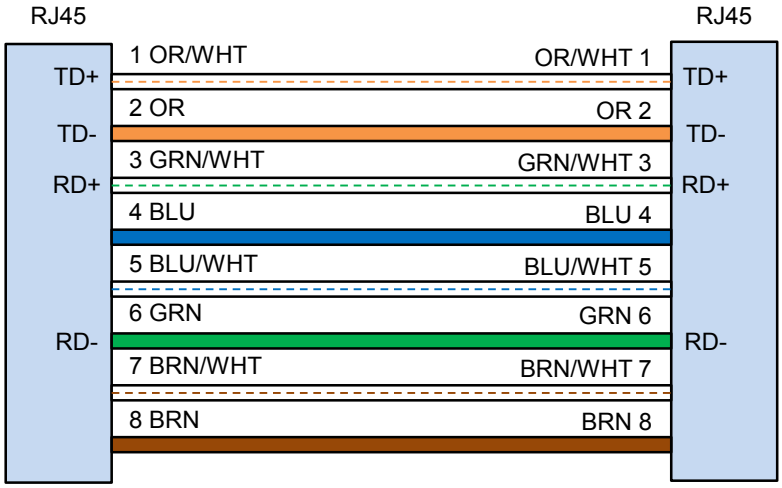


Figure 19: Straight-through cable scheme

3.2 USB3.0 Interface (UCC)

The USB3.0 interface provides configuration access to the camera and is also used for image data transmission. The connector is a standardized USB3.0 micro-B jack, the recommended mating connector is a standard Micro-B USB3.0 connector with screw lock.

3.2.1 Cabling Requirements

To connect the camera to a PC, a USB3.0 A to micro-B cable is required. As USB3.0 is a very compact high-speed interface, it is highly recommended to only use industrial grade cabling assembled with high-grade connectors providing appropiate locking to the camera port. The ratio between high-frequency characteristics and cooper wire gauge results into a maximum possible cable length. To avoid EMI, it is recommendet to use only shielded cables, a close installation to high frequency electromagnetic fields should be avoided.

Note

Bad cabling, connectors and/or shielding can lead to decreased performance (e.g. framerates) up to connection interrupts. It is thus highly recommended to purchase the right industrial cabling from our local SMARTEK Vision distribution partner to prevent issues in performance.

3.3 Power and I/O Interface

Beside the Ethernet and USB3.0 interface for communication and data transmission, all cameras are equipped with an *I/O-interface and power input*. Via this interface the cameras provide access to one opto-isolated digital input and one opto-isolated output. GCC cameras have additionally a dedicated power supply input.

Model	Connector Type	Receptable
GCC (standard housing)	6-pin Circular Hirose	HR10-7P-6S
UCC (standard housing)	6-pin Circular Hirose	HR10-7P-6S

Table 12: Power and I/O-interface connector type per model

6-pin Circular Hirose Connector

The housed twentynine standard cameras are equipped with a 6-pin circular Hirose receptacle to provide access to the power interface as well as the input and output lines. Figure 20 shows the pin and connector orientation on the back of the camera housing, Table 13 shows the corresponding pin assignment.

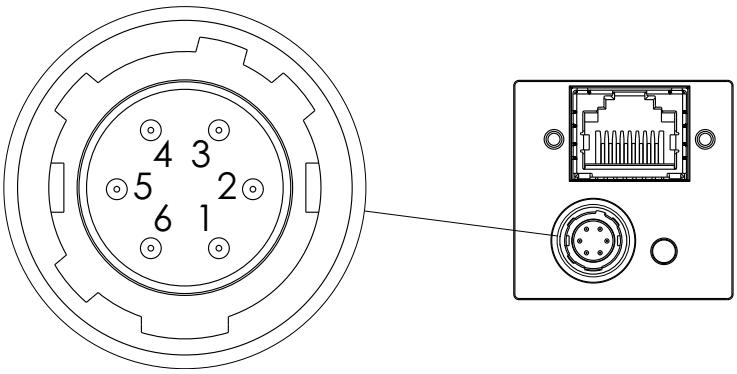


Figure 20: 6-pin circular Hirose receptacle - Pin and connector orientation

Pin no.	GCC Signals	UCC Signals
1	DC power supply	Direct coupled I/O (optional)
2	Opto-isolated In	Opto-isolated In
3	Direct coupled I/O (optional)	Direct coupled I/O (optional)
4	Opto-isolated Out	Opto-isolated Out
5	GND for opto-isolated I/O	GND for opto-isolated I/O
6	Power GND	GND for direct coupled I/O (optional)

Table 13: 6-pin circular Hirose receptacle - Pin assignment

Note The 6-pin connector on the camera is a Hirose receptacle and can be used with a HR10-7P-6S or equivalent.

3.3.1 Cabling Requirements

A single 6-pin Hirose receptacle is used to power the camera and provide access to its input and output lines. When assembling the 6 pin Hirose connector on one side of the cable, it must be taken care to follow the pin arrangement shown in Table 13 (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.

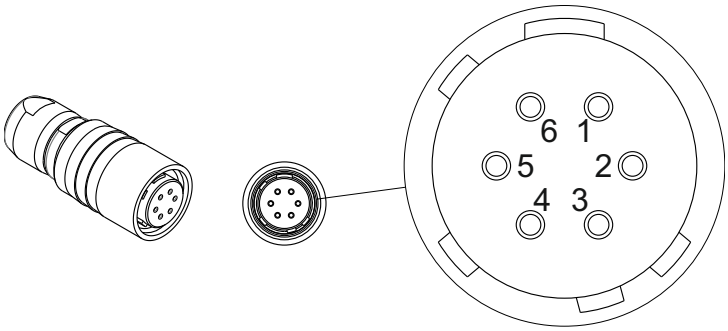


Figure 21: Hirose 6-pin plug connector

Note The 6 pin connector for the cable is a Hirose plug HR10-7P-6S(73) (or equivalent).

Note Close proximity to strong high-frequency electromagnetic fields should be avoided. It is recommended to use shielded twisted pair wires to ensure that signals are correctly transmitted.

An incorrect pin assignment or connector can lead to permanent damages to the camera.

3.3.2 Input Lines (Electrical Specification)

The cameras are equipped with a physical input line, designated as *opto-isolated in*. It is opto-isolated and can be accessed via the power and I/O interface receptacle on the back of the device. Table 14 shows the operational limits of the input line, while Figure 22 shows its 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 14: Electrical specification for trigger input (operational limits)

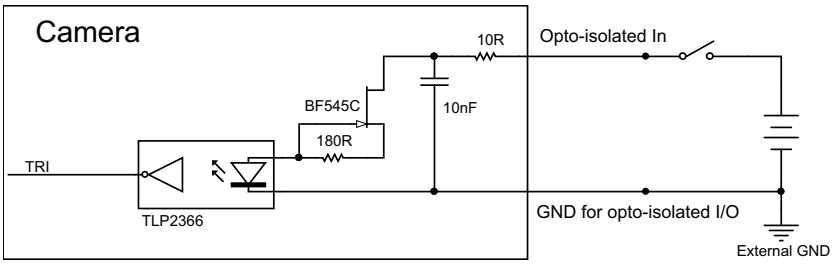


Figure 22: Trigger input scheme

Caution Exceeding the limits shown in Table 14 or reneging the wiring polarity shown in Figure 22 can seriously damage the device!

3.3.3 Output Lines (Electrical Specification)

The cameras are equipped with a physical output line, designated as *opto-isolated out*. It is opto-isolated and accessed via the power and I/O interface receptacle. Table 15 shows the operational limits of the output line, Figure 23 shows its electrical scheme.

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 15: Electrical specification for digital output

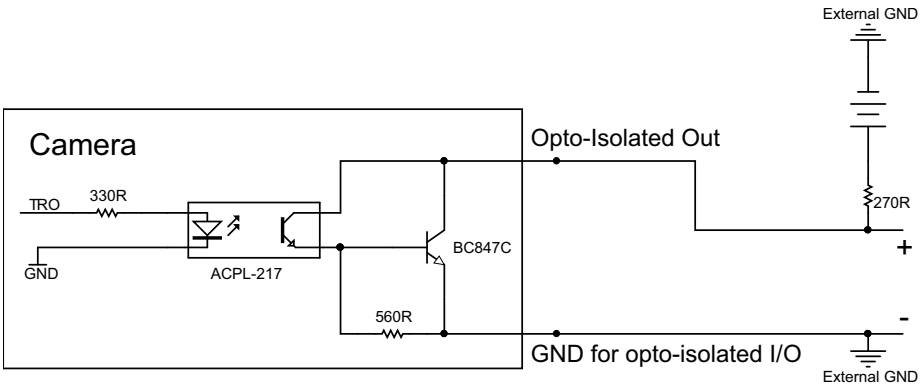


Figure 23: Digital output scheme

Caution Exceeding the limits shown in Table 15 or reneging the wiring polarity shown in Figure 23 can seriously damage the device!

3.4 Status LED

All cameras have a bi-color status LED on the back of their housing. During boot-up, it will steadly output yellow light for several seconds. As soon as the camera is operational, it switches to steady green light. Errors are indicated by a specific number of red pulses on the status LED, described in Table 16.

Status LED	Description
Steady Green	Camera is operational (status OK)
Steady Yellow	Camera is powered on and booting up
Blinking Red	Camera is in error state - number of pulses indicate the detected error: • 1 pulse - no user firmware present • 2 pulses - user firmware watchdog timeout • 3 pulses - user firmware data CRC error • 4 pulses - error configuring FPGA (NSTATUS asserted)

Table 16: Status LED

4 General Camera Architecture

The twentynine camera series consist of multiple electronic components to convert incoming light to a digital signal, process it and send it to the host device. Figure 24 shows the simplified architecture of the twentynine camera, consisting of the image data path (orange) as well as multiple control paths (red).

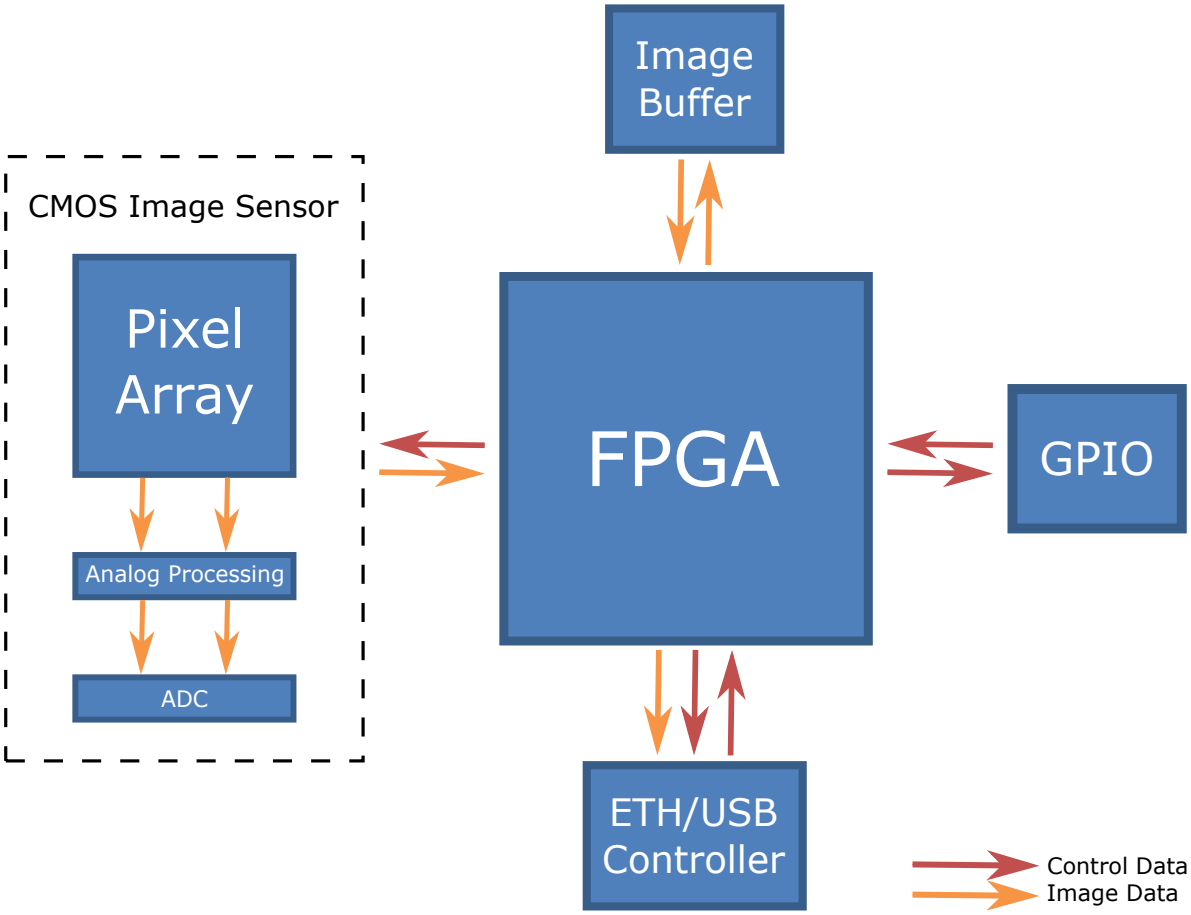


Figure 24: Camera Architecture Scheme

The image data path starts with the image sensor. When photons hit sensor's active area during the integration time, electrical charge accumulates in pixels. After the integration time has ended, accumulated charges are transferred from the light sensitive elements to the *Analog Processing* block where they are converted to voltages proportional to accumulated charges. The voltages are then amplified and finally digitized by an *Analog to Digital Converter* (ADC). Digitized pixel data leaves the sensor and enters the FPGA which performs image processing on pixel data and stores the resulting image in the image buffer.

The last step of the image data path is the transmission via the *Transport Layer Controller* (Ethernet / USB). The image data is segmented into GigE Vision / USB3 Vision packets and transmitted to the host computer.

The *Transport Layer Controller* also handles transmission and reception of control data which are processed by the FPGA.

4.1 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.

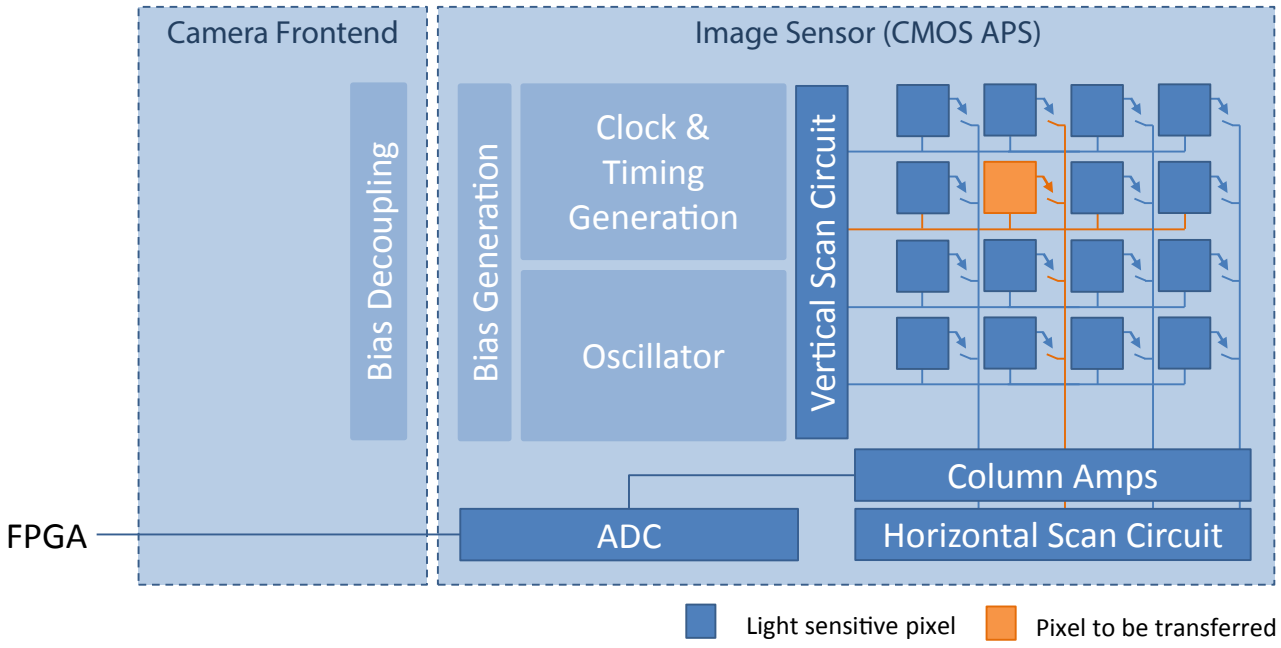


Figure 25: twentynine Frontend with CMOS Active Pixel Sensor with integrated Gain and ADC

As shown in Figure 25, 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^2C / SPI).

4.2 Color Imaging with Bayer Pattern

In an area image sensor pixels are arranged in a two dimensional array (see Figure 26). 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 twentynine family are equipped with area image sensors with Bayer pattern.

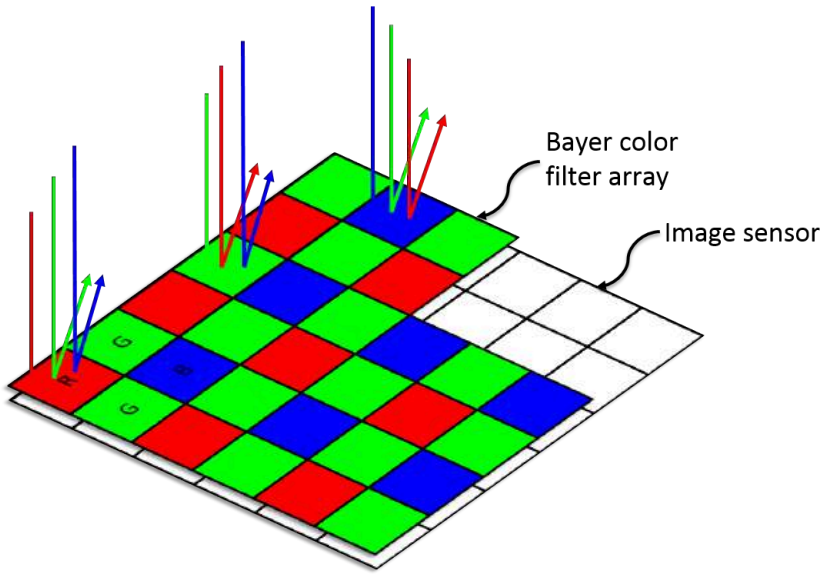


Figure 26: Bayer Color Filter Array placed on top of an area image sensor

Figure 26 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 26 is therefore called a "RG" pattern. This pattern is one of the four Bayer patterns available: GR, RG, BG and GB shown in Figure 27.

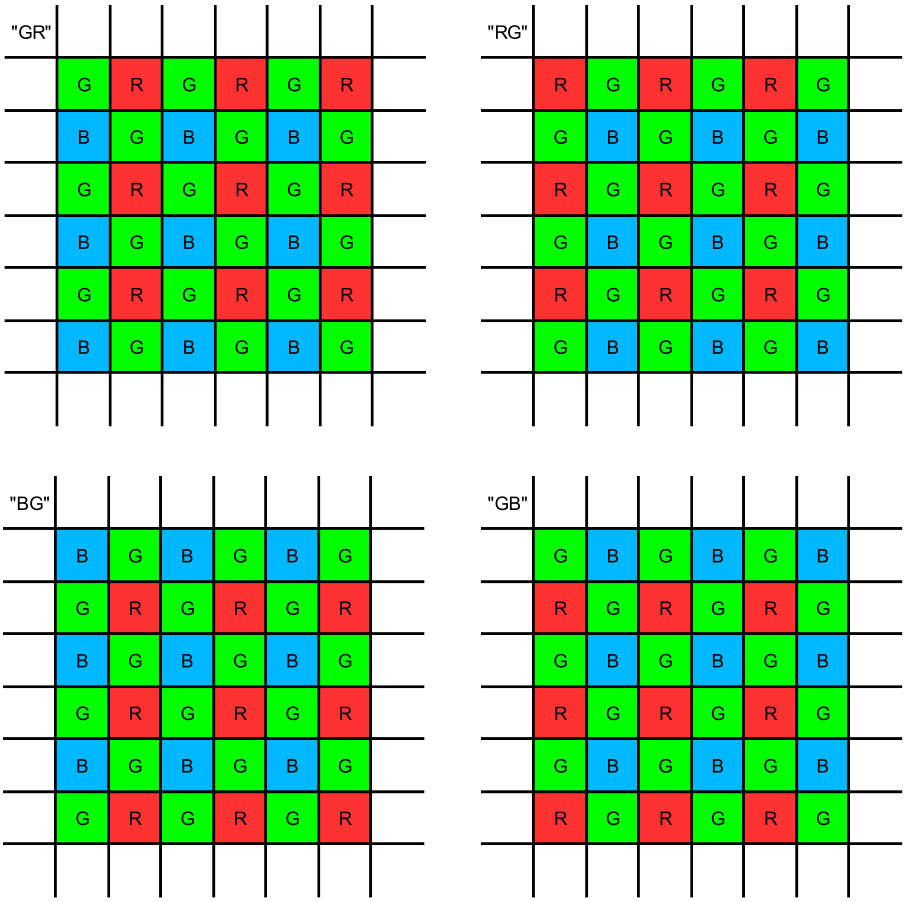


Figure 27: Bayer Color Filter Array placed on top of an area image sensor

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 *8.2.6 - Color Filter Array Interpolation (Demosaicing / Debayering)* in this user manual.

4.3 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.

4.3.1 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 28; 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.

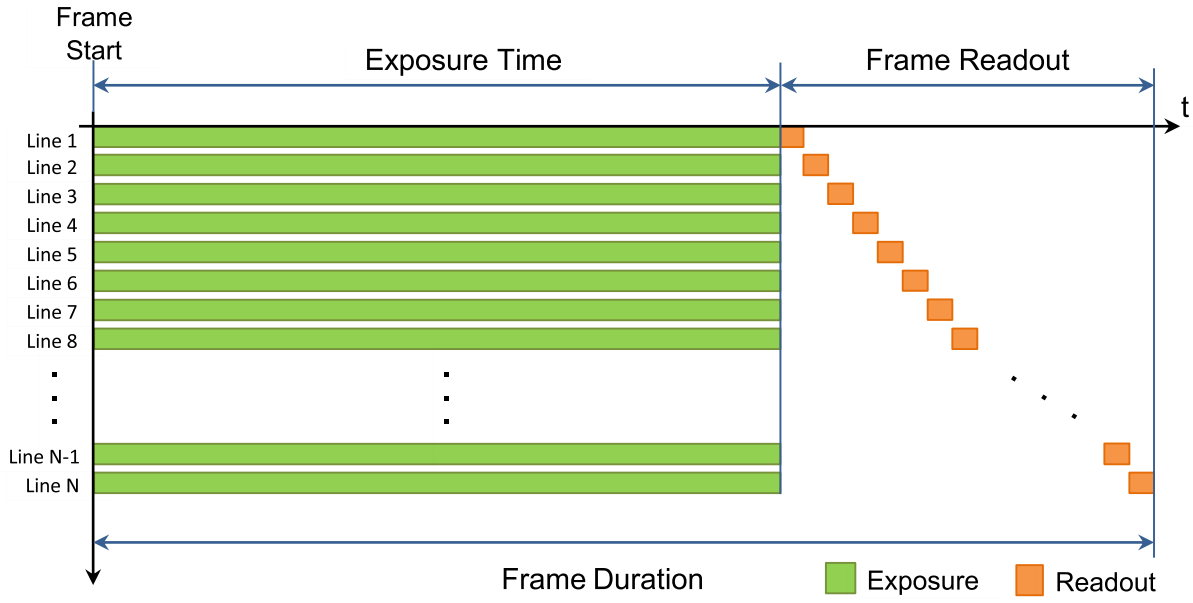


Figure 28: Global Shutter Frame Readout

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.

4.3.2 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 29; 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 $t_{ReadRow}$ to the prior line as well. Beside some internal timing parameters and the read out frequency, $t_{ReadRow}$ is mainly affected by the image width. The total time for frame read out ($t_{FrameReadout}$) can be calculated by multiplying $t_{ReadRow}$ with the total count of lines in the frame.

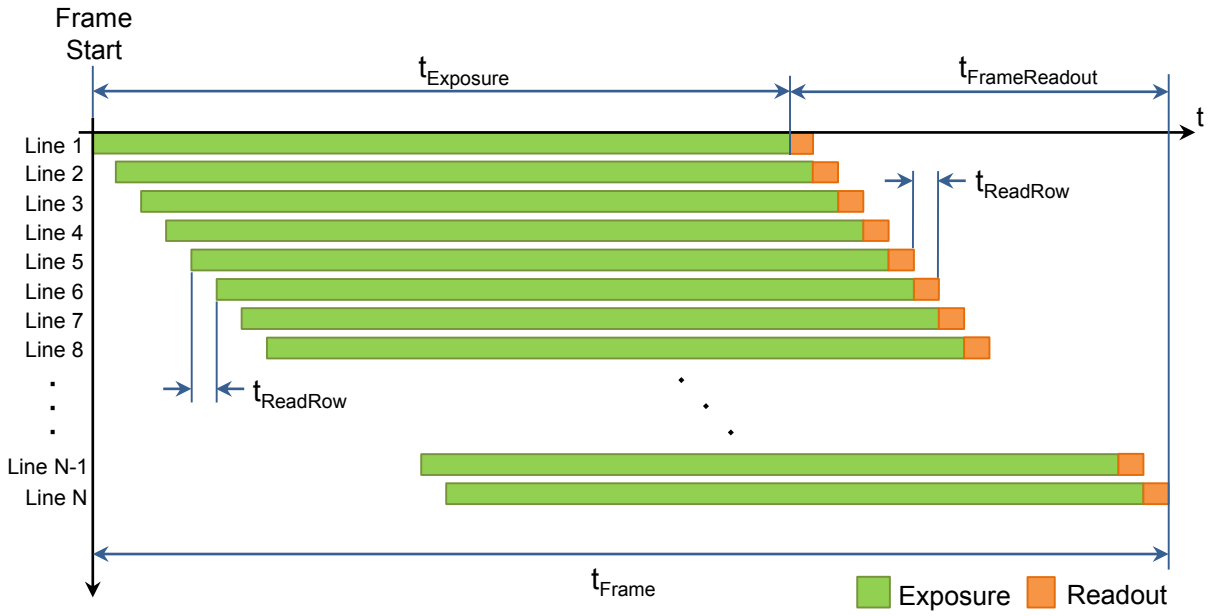


Figure 29: Electronic Rolling Shutter Frame Readout

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:

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

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.

4.3.2.1 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 30 shows this illumination window as $t_{Illumination}$, staring at $t_{IlluminationDelay}$.

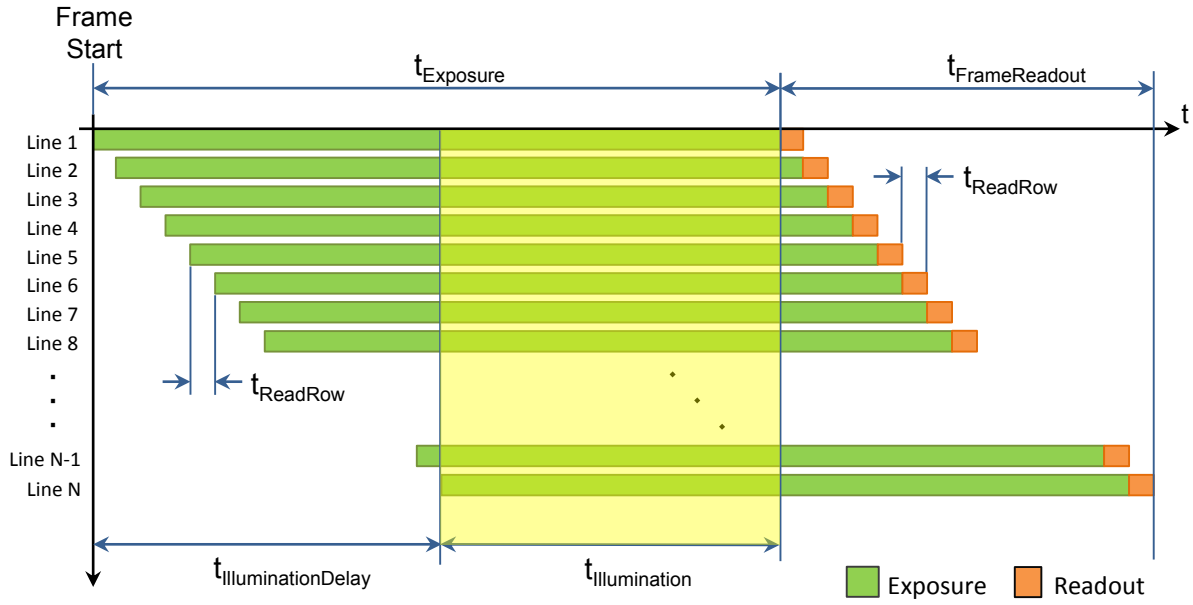


Figure 30: Electronic Rolling Shutter Frame Readout

Beyond the illumination period $t_{Illumination}$, 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:

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

On time of illumination / shutter open:

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

4.3.3 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 31, 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 $t_{ReadRow}$ this way to its prior, the image lightens up line by line from top to bottom.

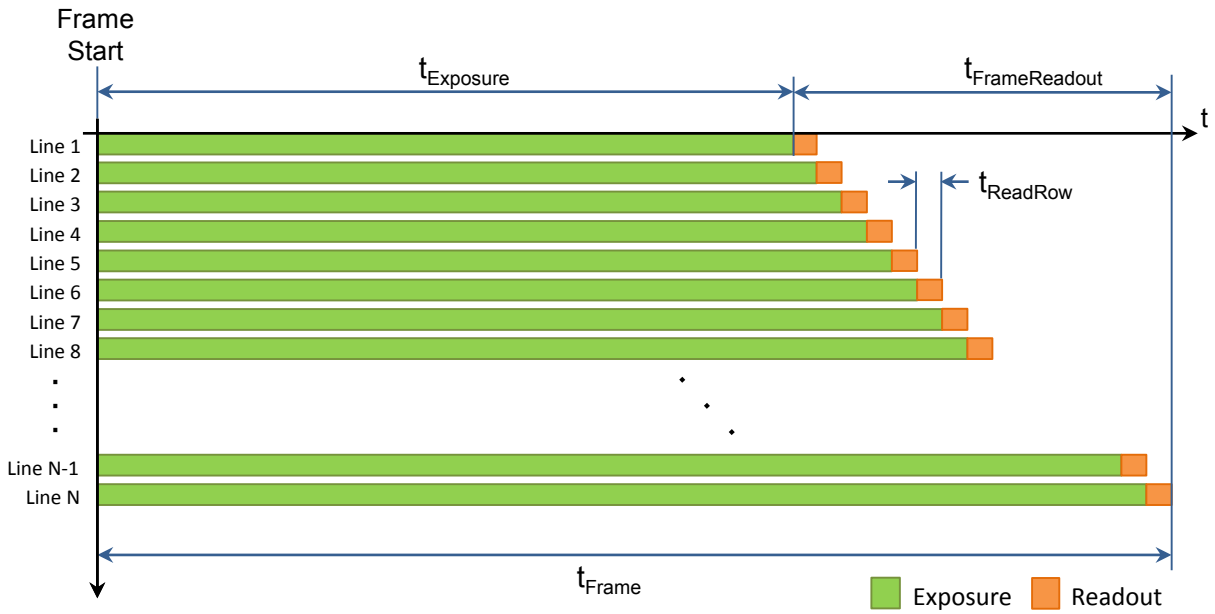


Figure 31: Global Reset Release (GRR) Frame Readout

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.

5 Camera Features

The following chapter gives a brief overview about the general principles of digital image acquisition based on the twentynine 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.

5.1 List of Supported Features

Series	twentynine	
Interface	GiGE VISION	USB3 VISION
On-Camera Features	**GCC**	**UCC**
Continuous Streaming (free run)	●	●
Triggered Operation (single / multi frame)	●	●
Exposure Control	●	●
Auto Exposure Control	●[1]	●[1]
Frame Rate Control	●	●
Partial Scan (ROI / AOI / WOI)	●	●
Multiple ROI	●	●
ROI Centering	●	●
Binning	●[2]	●[2]
Reverse X (Horizontal Mirroring)	●	●
Reverse Y (Vertical Mirroring)	●	●
Analog Gain Control	●	●
Auto Analog Gain Control	●[1]	●[1]
Analog Black Level Control	●	●
Gamma	●[1]	●[1]
Lookup Table	●[1]	●[1]
Digital Shift	●	●
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	●	●

Table 17: Camera feature list (1/2)

Series	twentynine	
Interface	GiGE VISION	USB3 VISION
On-Camera Features	**GCC**	**UCC**
IP Configuration (LLA / DHCP / Persistent)	●	○
Jumbo Frame Size (in Bytes)	8000	○
Inter Packet Delay	●	○
Frame Transfer Delay	●	○
Time Stamps	●	●
Pixel Data Formats		
Mono8	●[1]	●[1]
Mono10Packed	○	●[1]
Mono12Packed	●[1]	○
Bayer8	●[3]	●[3]
Bayer10Packed	○	●[3]
Bayer12Packed	●[3]	○

1 Mono models only
2 Horizontal (mono models only)
3 Color models only

Table 18: Camera feature list (2/2)

5.2 Brightness and Sensor Signal Control

5.2.1 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 32 demonstrates two settings of camera's exposure time. The picture on the left is captured with an exposure time of 10000 µs. For the picture on the right the exposure time is set to 22000 µs. 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 32: Different exposure time settings

The exposure time for SMARTEK Vision digital cameras is configurable by the GenICam *Float* property ExposureTime and is expressed in microseconds (µs). Each camera has a predefined range of values, depending on the sensor and its technology.

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

Table 19: ExposureTime - Access through API

Table 19 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 CameraSuite SDK 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 µs 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 CameraSuite SDK. For detailed description of the automatic exposure feature please refer to chapter *8.2.3 - Auto Exposure and Auto Gain*.

5.2.2 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.

5.2.2.1 Analog Gain

Figure 33 illustrates a typical image acquisition signal chain in SMARTEK 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.

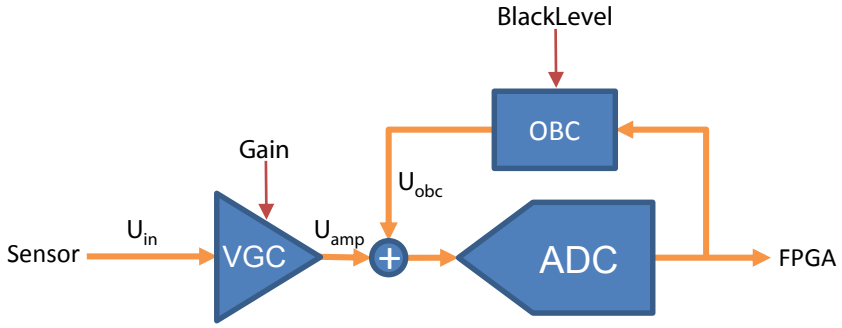


Figure 33: Typical signal chain in a CCD/CMOS image sensor

In SMARTEK 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:

$$\mathrm{Gain_{dB}} = 20 \times \log_{10} \frac{\mathrm{U_{amp}}}{\mathrm{U_{in}}}$$

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

$$\frac{\mathrm{U_{amp}}}{\mathrm{U_{in}}} = 10^{\frac{\mathrm{Gain_{dB}}}{20}}$$

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 34 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 34: 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 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 20.

Function	Description
bool GetEnumNodeValue ("GainSelector", double &nodeValue) const	Get value of Enumeration node GainSelector.
bool SetEnumNodeValue ("GainSelector", double nodeValue)	Set value of Enumeration node GainSelector.
bool GetEnumNodeValuesList ("GainSelector", StringList &nodeValuesList) const	Get list of values for Enumeration node GainSelector.

Table 20: GainSelector - Access through API

The values for the Enumeration data type *GainSelector* can be found in Table 21, their availability depends on the camera architecture and can be requested from the camera as shown in Table 20.

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

Table 21: GainSelector - Values

After the appropriate gain has been selected via the *GainSelector*, its value can be get/set from the *Gain* property. Table 22 shows the most important C++ functions.

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

Table 22: Gain - Access through API

5.2.2.2 Black Level

As shown in Figure 33 as well, the analog-to-digital conversion circuit includes beside the ADC an additional component with the target to remove dark current from the signal of each pixel. Dark current represents a number of charges generated in each pixel due to thermical processes inside the silicon lattice, even when no light enters the pixel. 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 component, 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 33, 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 Current subtracted from the signal by adding a user defined offset to the overall signal.

5.2.3 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 twentynine 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 23.

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

Table 23: 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
TargetGrayValue	Integer	The average image brightness that should be reached
ExposureTimeAbsLowerLimit	Float	The minimum exposure time (in µs) that is allowed to be set by the *Automatic Exposure* algorithm
ExposureTimeAbsUpperLimit	Float	The maximum exposure time (in µs) that is allowed to be set by the *Automatic Exposure* algorithm
GainAbsLowerLimit	Float	The minimum gain value (in dB) that is allowed to be set by the *Automatic Gain* algorithm
GainAbsUpperLimit	Float	The maximum gain value (in dB) that is allowed to be set by the *Automatic Gain* algorithm

Table 24: 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 CameraSuite SDK. For detailed description of the automatic exposure and gain features in the SDK please refer to chapter *8.2.3 - Auto Exposure and Auto Gain*.

5.2.4 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 (usually 10, 12 or 14 Bits) to 8 Bit. It is available for Mono8 or Bayer8 pixel formats. As shown in Figure 35, all cameras by default use the 8 Most Significant Bits (MSB) of the Analog-to-Digital Converter (ADC) to create 8 Bit pixels, the Least Significant Bits (LSB) are cut off.

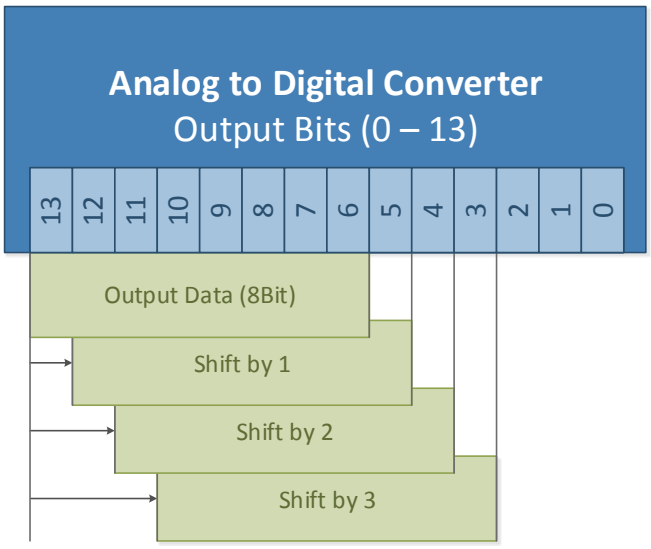


Figure 35: Digital Bit Shifting at Analog to Digital Converter

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 25 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 25: 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.

5.2.5 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 8.2.5 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.

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

Users can change the gamma value under *Analog Control* property like shown in Figure 36. 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*.

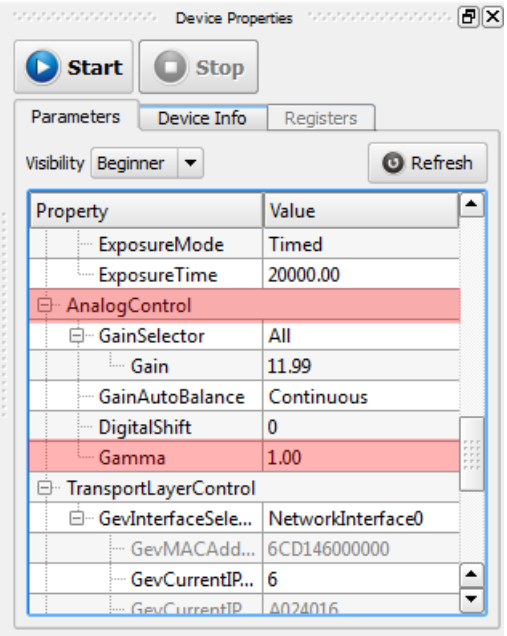


Figure 36: Modify gamma value in CameraSuiteClient

5.2.6 Luminance Look-up-Table

The twentynine camera series equipped with monochrome sensors support 12bit look-up table. The theory and applications of look-up table will be introduced in chapter 8.2.1.

Figure 37 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* ⇒ *Image Processing Properties*.

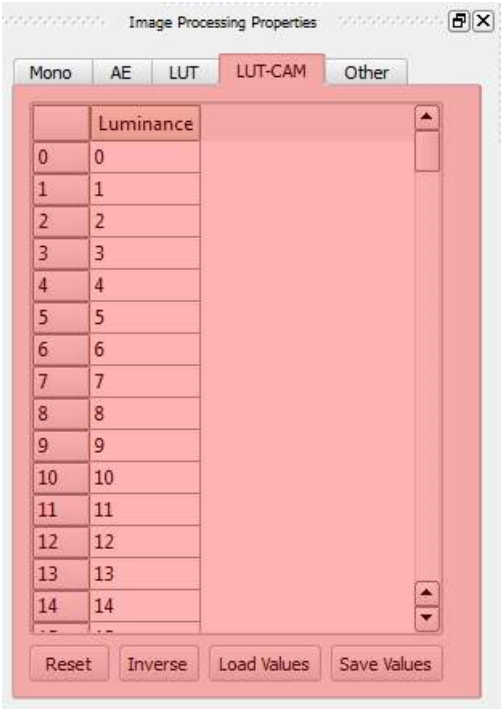


Figure 37: LUT-CAM feature in CameraSuiteClient

- **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 *CameraSuiteClient* (see Figure 37). Please refer to the Look-up table section in chapter 8.2.1 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 *Visibilty* option in *Device Properties* panel to *Expert* or *Guru* (see Figure 38 and Figure 39).

In order to modify individual LUT value it is neccessary that the *Visibilty* 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 39).

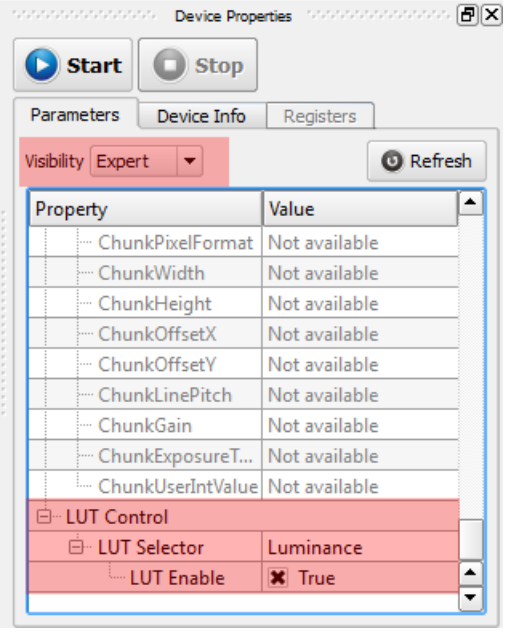


Figure 38: Enable LUT feature on Camera in CameraSuiteClient

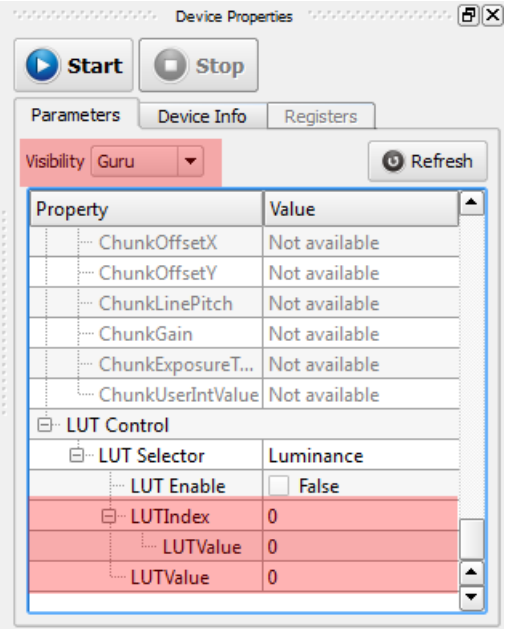


Figure 39: Modify individual LUT value in CameraSuiteClient

5.3 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.

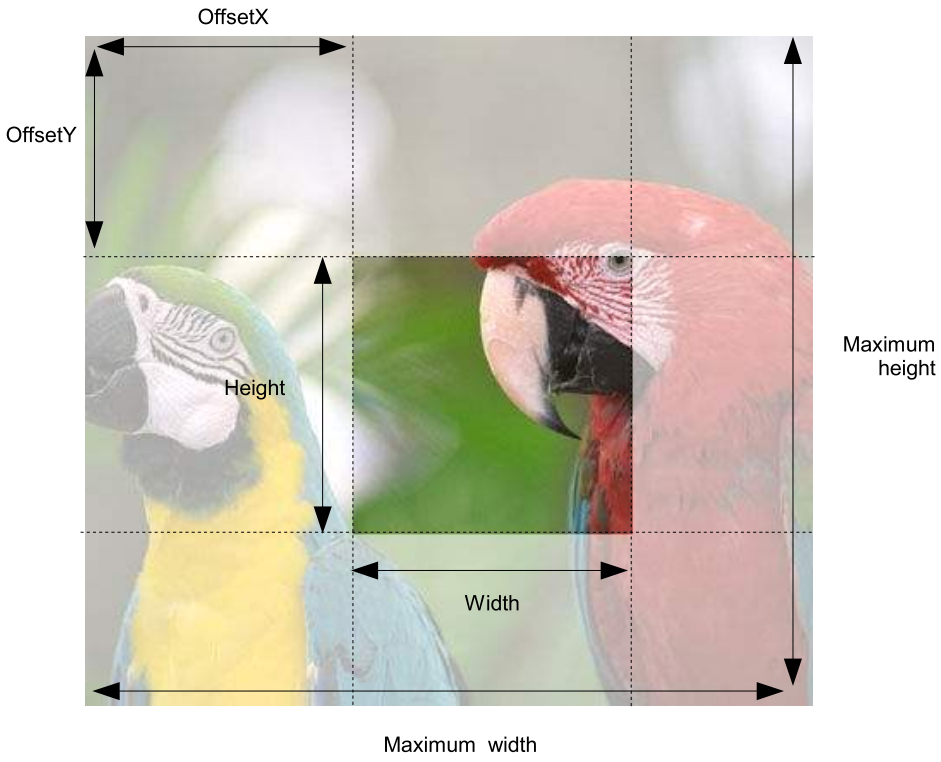


Figure 40: Region of Interest

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
Width	Integer	Horizontal size of the ROI image (in pixels)
Height	Integer	Vertical size of the ROI image (in pixels)
OffsetX	Integer	Horizontal offset from the origin to the ROI (in pixels)
OffsetY	Integer	Vertical offset from the origin to the ROI (in pixels)

Table 26: 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.

5.3.1 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
RegionSelector	Enumeration	Selects the Region of Interest to control
RegionMode	Boolean	Enables or Disables the selected region
RegionDestination	Enumeration	Controls the destination stream of the selected region

Table 27: Multiple Regions of Interest parameters

Each ROI defines one horizontal and one vertical stripe in the full resolution image as shown in Figure 41. 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.

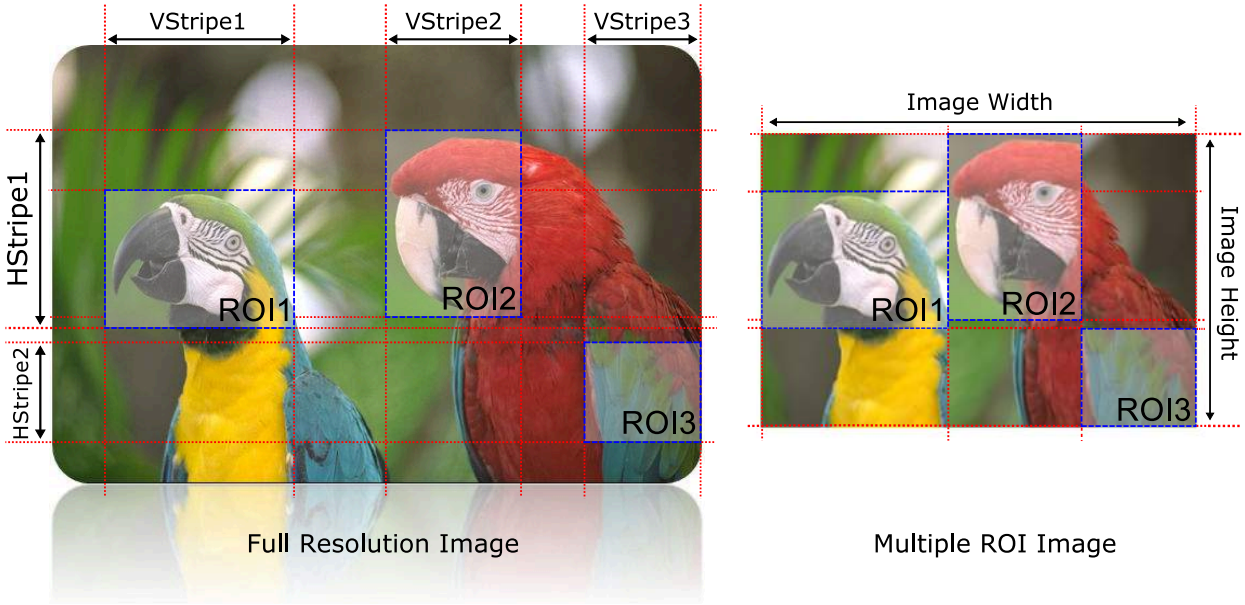


Figure 41: Multiple Regions of Interest

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 41 are listed in Table 28 below. It shows the definition of all three ROIs.

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

Table 28: 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*.

5.3.2 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.

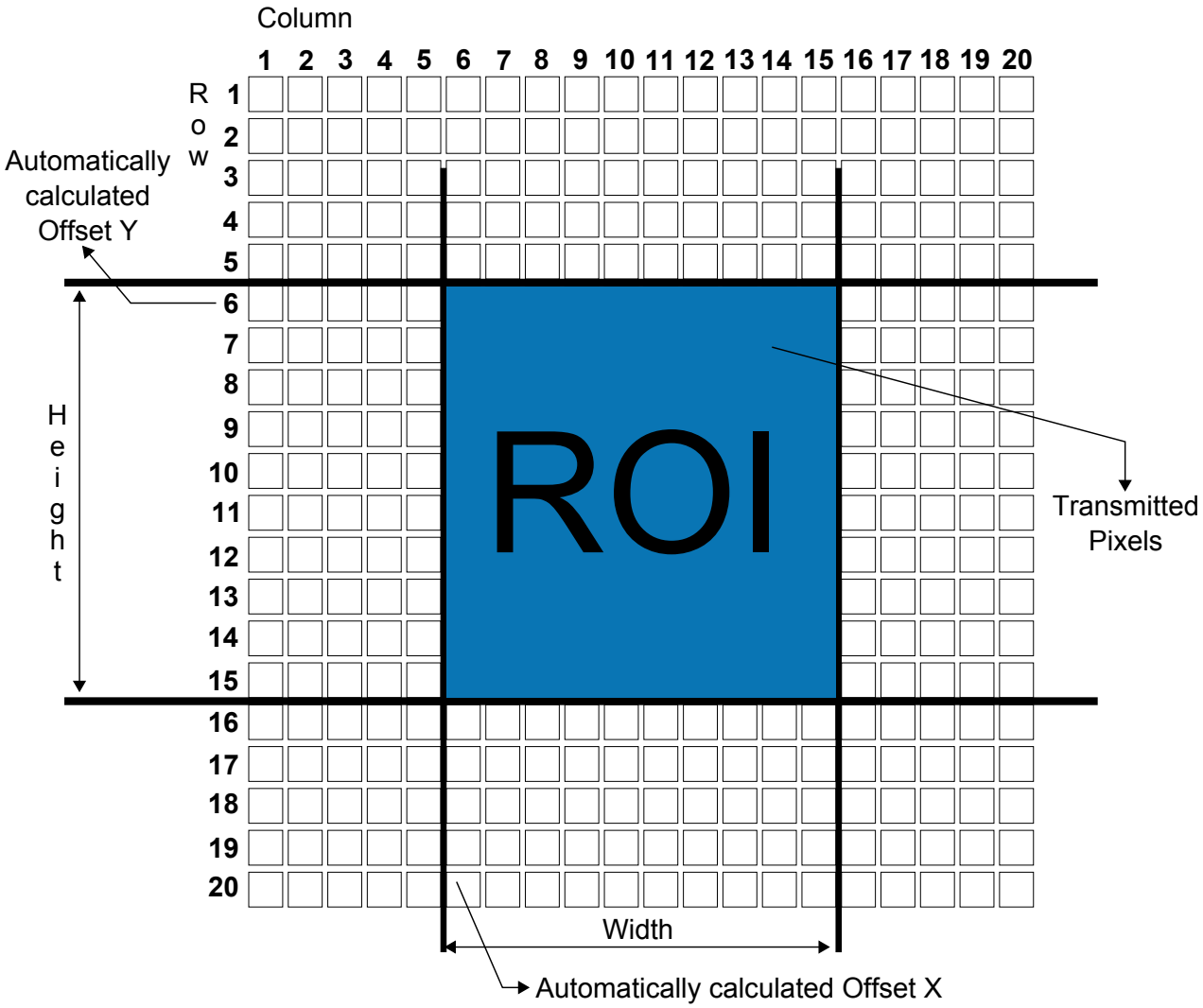


Figure 42: Center X & Y

5.4 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 29 gives a brief overview about all features that are available to control the image acquisition in the cameras.

Image Acquisition Features	Short description
AcquisitionMode	Defines the number of frames to be captured. Three options are available: • Continuous • SingleFrame • MultiFrame
AcquisitionStart	Start acquisition
AcquisitionStop	Stop acquisition
AcquisitionAbort	Abort acquisition
AcquisitionFrameCount	Number of frames to acquire in MultiFrame acquisition mode
AcquisitionBurstFrameCount	Number of frames to acquire for each FrameBurstStart trigger
AcquisitionFrameRate	Controls the acquisition rate (in Hz) at which the frames are captured
Trigger Features	**Short description**
TriggerMode	Enable/Disable the trigger mode. Two options are available: • On • Off
TriggerSoftware	Generate a software trigger
TriggerSource	Select the source that fires a trigger signal: • Line1: Physical Input Line 1 • Line2: Physical Input Line 2 • Software
TriggerActivation	Define the clock edge of the input signal for activate triggering • Rising Edge • Falling Edge
TriggerDelay	Specify the delay in microseconds (μs) to incoming trigger signals.

Table 29: Camera features for image acquisition control

5.4.1 Image Acquisition Features

The *Image Acquisition Features* chapter describes features related to image acquisition. An *Acquisition* is defined as the capture of a sequence of one or many *Frames* or *Images*. The *Frames* of an *Acquisition* can optionally be grouped in smaller *Bursts* that are triggered individually.

5.4.1.1 Acquisition Mode

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 30.

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

Table 30: AcquisitionMode values

5.4.1.2 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 bandwidth required by the camera.

This feature is useful in situations where the link bandwidth is limited, like in applications where several GigE Vision cameras stream images to PC via switch and there is single Gigabit Ethernet link connecting switch and NIC on PC.

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

5.4.2 Trigger Features

The *Trigger Features* chapter describes features related to synchronizing the camera with an external device or software application.

5.4.2.1 Trigger Selector

The *TriggerSelector* property provides access to the properties of different triggers supported by the camera, as shown in Table 31.

Value	Description
AcquisitionStart	A trigger that starts the acquisition of one or many frames according to *AcquisitionMode*
FrameStart	A trigger starting the capture of one frame
FrameBurstStart	A trigger starting the capture of the burst of frames in an acquisition. *AcquisitionBurstFrameCount* controls the length of each burst. The total number of frames captured is also conditioned by *AcquisitionFrameCount* when *AcquisitionMode* is *MultiFrame*.

Table 31: Supported Trigger types

5.4.2.2 Trigger Mode

Selected trigger can be enabled or disabled via the *TriggerMode* property.

Value	Description
On	Enables the selected trigger
Off	Disables the selected trigger

Table 32: Trigger Mode

5.4.2.3 Trigger Source

The *TriggerSource* property specifies the internal signal or physical input line to use as the trigger source. The selected trigger must have its *TriggerMode* set to *On*. Possible values for the *TriggerSource* are shown in Table 33.

Value	Description
Line1	Specifies that the physical Input Line1 will be used as an external source for the trigger signal
Software	Specifies that the trigger source will be generated by software using the *TriggerSoftware* command

Table 33: Trigger Sources

5.4.2.4 Trigger Activation

For some trigger sources it is possible to define the activation mode of the trigger. Possible values are presented in Table 34.

Value	Description
RisingEdge	Specifies that the trigger is considered valid on the rising edge of the source signal
FallingEdge	Specifies that the trigger is considered valid on the falling edge of the source signal

Table 34: Trigger Activation modes

Figure 43 shows the exposure period of the sensor triggered by a rising edge, while Figure 44 shows the exposure period of the sensor triggered by a falling edge of the source signal.

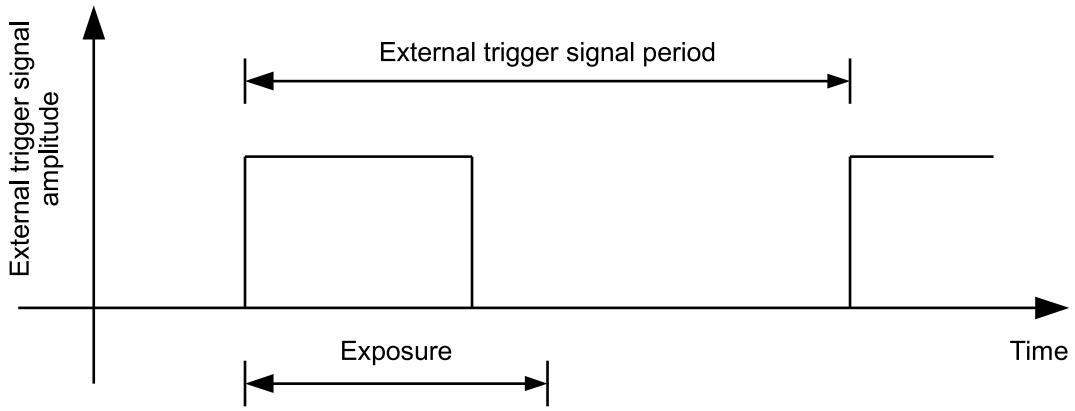


Figure 43: Exposure with a rising edge of the trigger

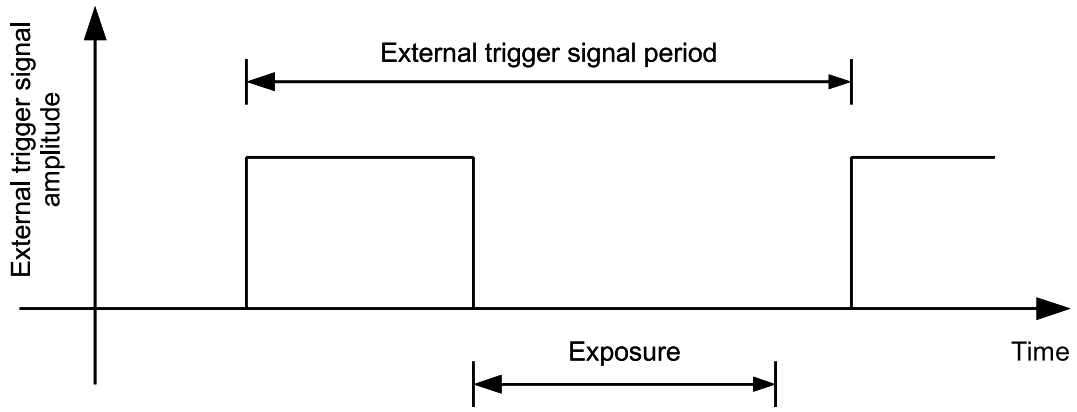


Figure 44: Exposure with a falling edge of the trigger

5.4.2.5 Trigger Delay

The *TriggerDelay* property specifies the delay in microseconds (µs) to apply after the trigger reception and before activating it internally.

A further description of the complete trigger process can be found in *5.5.1 - Input Lines*.

5.4.3 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 fixed frequency, which is the maximum possible by default but can be limited via the *AcquisitionFrameRate* parameter.
In order to enable the *Free Run* mode, all triggers must be disabled (*TriggerMode* set to *Off*) and *Acquisition Mode* must be set to *Continuous*.

5.5 Digital Input / Output Control

The digital inputs and outputs of the twentynine 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 3.3 - Power and I/O-Interface.

Each physical line is configured separately and can be selected by the *LineSelector* property. The property *LineMode* provides information if the currently selected line is an *Input* or *Output*. Table 35 describes all available configuration properties.

Property	Type	Description
LineSelector	Enumeration	Select the line for configuration; all further properties contain the values based on the selected line. Values are: • Line1 • Line2
LineMode	Enumeration	Specifies if the selected line is an Input or Output
LineStatus	Boolean	Current status of the selected line
LineSource	Enumeration	Source event driving the output line
LineFormat	Enumeration	Electrical format of the selected line
LineDebouncerTime	Float	Define the debouncer time of input lines (in µs)

Table 35: Trigger Sources

5.5.1 Input Lines

The camera's input lines can be used to trigger an acquisition of the camera on external events. Figure 45 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 user defined *TriggerDelay* increased by the fixed trigger latency caused by internal circuitry defines the start of the triggered event.

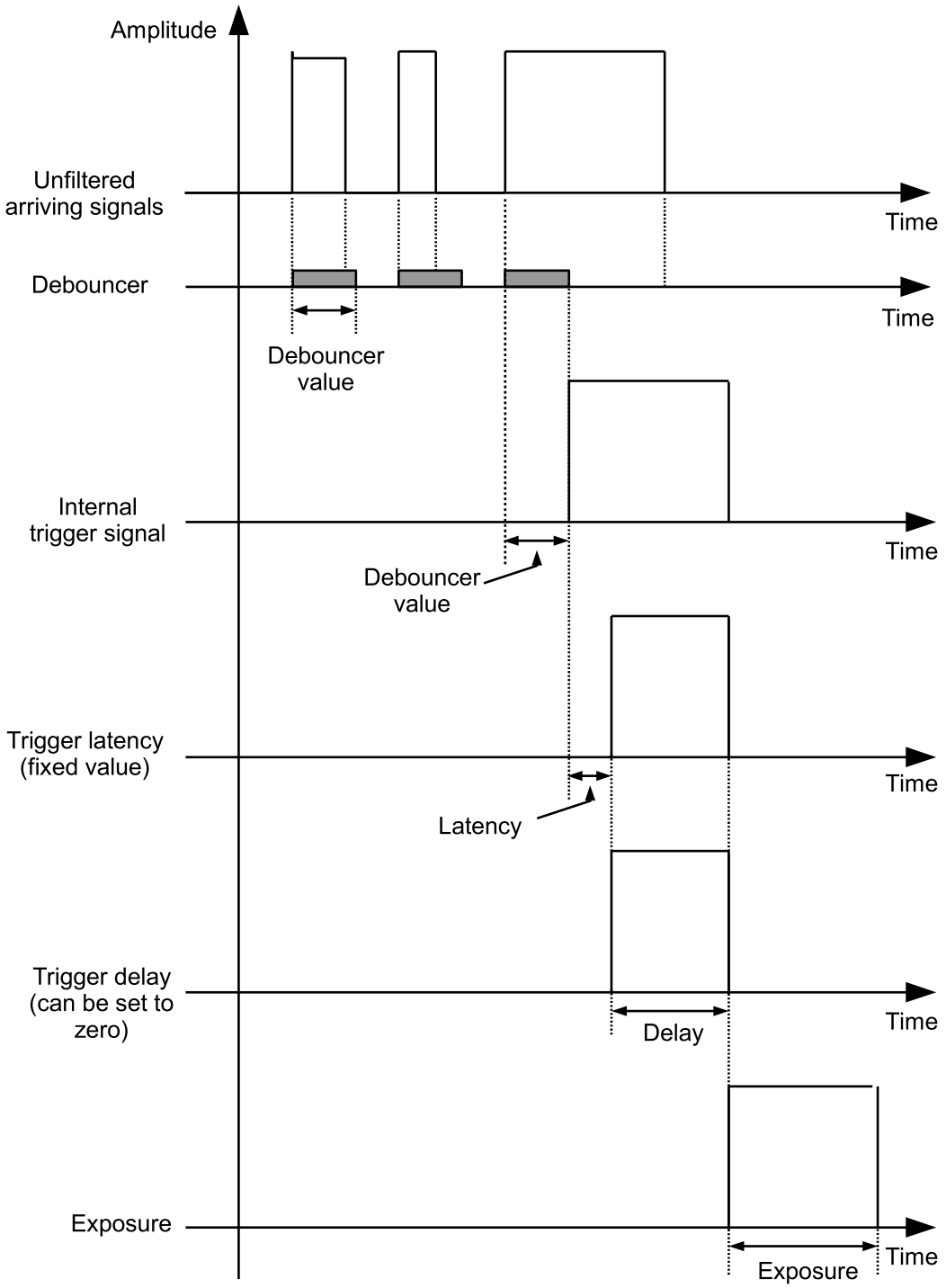


Figure 45: Partial process of image acquisition

5.5.1.1 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 46; 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 µs.

The line debouncer time effectively increases delay time between external trigger signal and internal trigger signal used to start the selected event, 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.

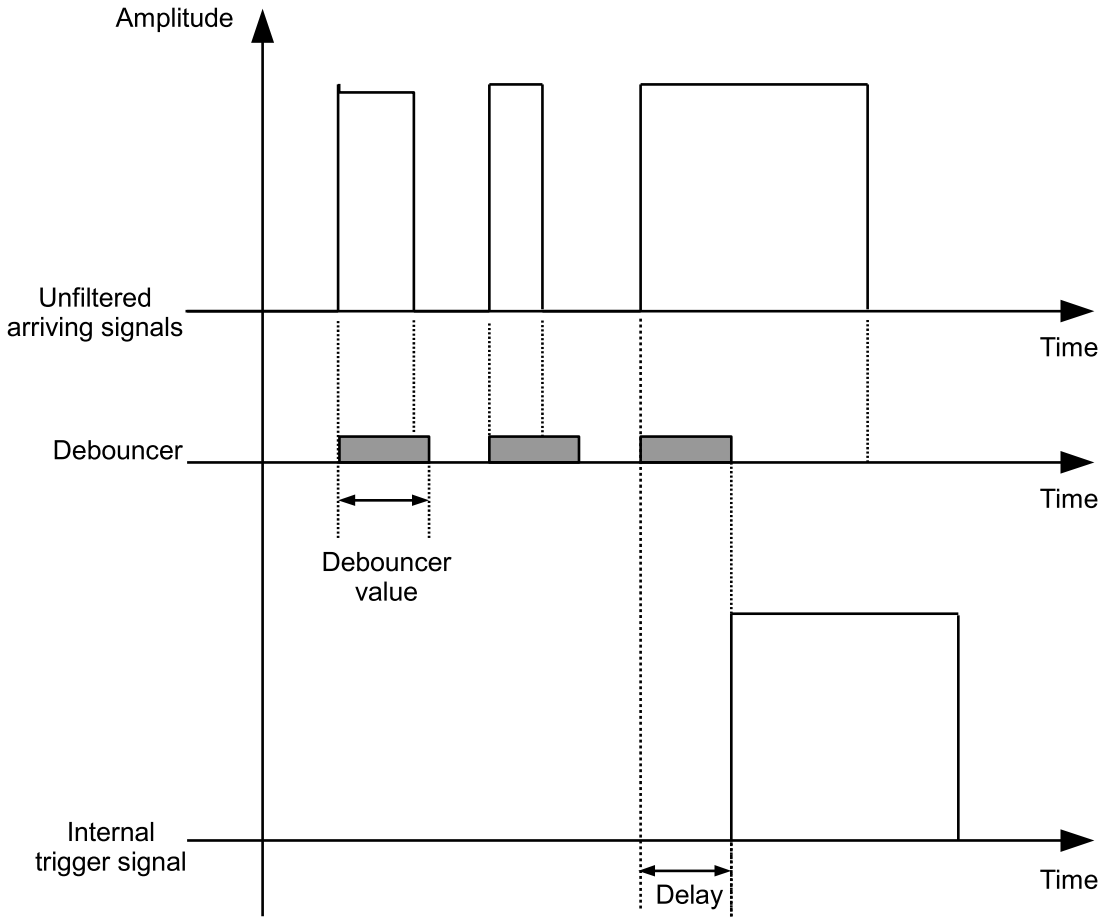


Figure 46: Line debouncer function

5.5.2 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 36.

Value	Description
ExposureActive	High while the exposure of a frame is active
UserOutput1	The state of the appropriate UserOutputValue property
UserOutput2	The state of the appropriate UserOutputValue property
AcquisitionTriggerWait	High while the camera waits for a trigger for one or more frames
AcquisitionActive	High while camera acquires one or more frames
FrameTriggerWait	High while camera waits for frame trigger
FrameActive	High while camera captures a single frame

Table 36: Output Line 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 37.

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

Table 37: User Outputs

5.6 GigE Vision Specific Features

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 twentynine 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.

5.6.1 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 38 shows the basic driver parameters available.

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

Table 38: Packet Resend Parameters (1/2)

Table 39 lists further parameters, allowing a detailed configuration of the *Packet Resent* mechanism. All this parameters mainly affect the performance and the robustness of the packet resending, changes should only be done carefully.

Parameter	Type	Description
PacketResendTimeout	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.
PacketResendResponseTimeout	Integer	Represents how long (ms) the Packet Resend Mechanism will wait for response after sending a resend request, until another resend request is sent.
MaxResendPacketRetry	Integer	Represents the maximum number of resend requests sent for a missing packet.
MaxMissingPacketWaiting	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.
MaxNextPacketWaiting	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.
MaxMissingPacketsCount	Integer	Maximum number of missing packets in one frame. If the frame has more missing packets then this value it will be dropped.
MaxNewImagesPending	Integer	Maximum amount of new images pending in the buffer. Current image is dropped if this amount is exceeded.
MaxNewPacketsPending	Integer	Maximum amount of incoming payload packets pending in the buffer. Current frame is dropped if this amount is exceeded.
MaxIncompletePackets	Integer	Maximum amount of missing payload packets from a block of the image to be accepted.

Table 39: 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 *7.1.3 - Network Interface Optimization*.

Example 1

Figure 47 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

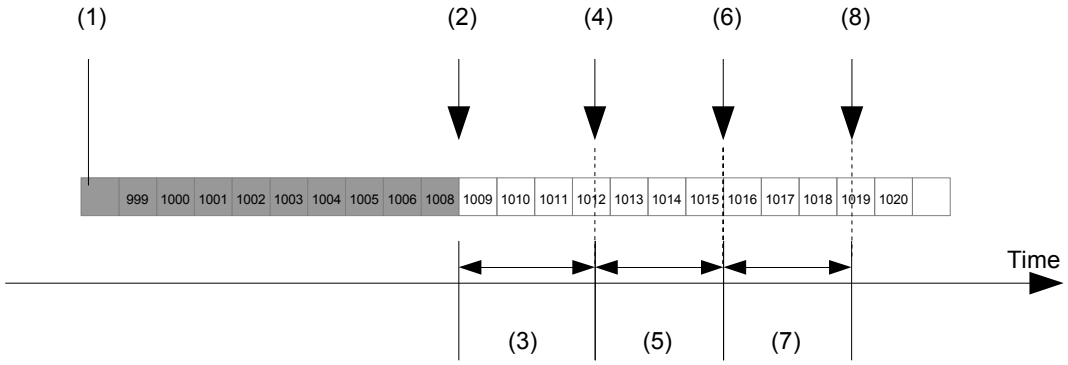


Figure 47: Packet Resend example when MaxResendPacketRetry value has exceeded

Corresponding to Figure 47, the workflow of the Packet Resent 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 48 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 resent request is sent.

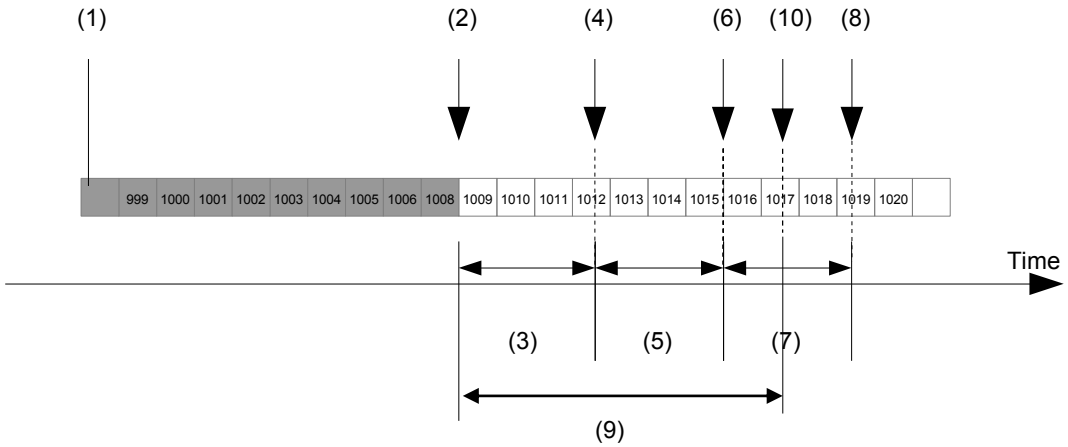


Figure 48: Packet Resend Mechanism example MaxMissingPacketWaiting value has exceeded

Additionally to the description in Example 2, the workflow of the Packet Resent 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.

5.6.2 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.

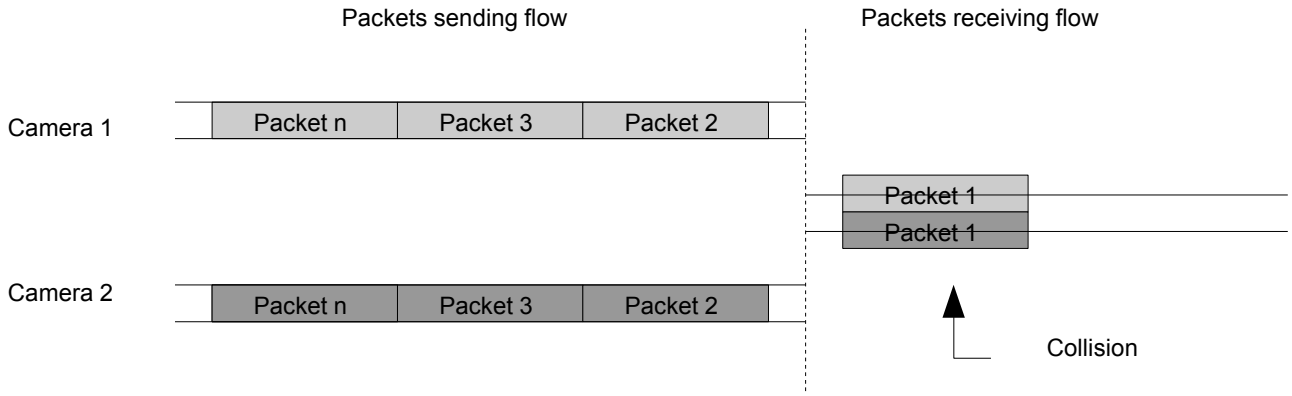


Figure 49: Packet flow while not using inter packet delay

If the *Inter Packed Delay* is not used, excessive collision between packets may occur which results in data loss, like illustrated in Figure 49. 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 50 illustrates a well configured *Inter Packet Delay* to prevent collisions.

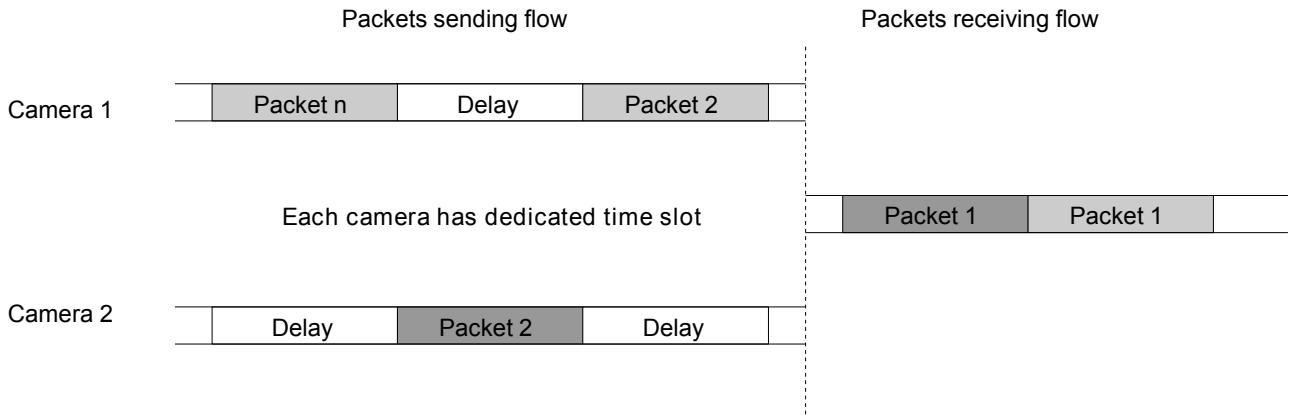


Figure 50: Packet flow while using inter packet delay

5.6.2.1 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 1000µs 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 12,3µs. 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 16,67µs 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 25µs (12,3µs + 12,3µs ≈ 25µs) will ensure that packets from other two cameras will fit in the gap between two consecutive packets.

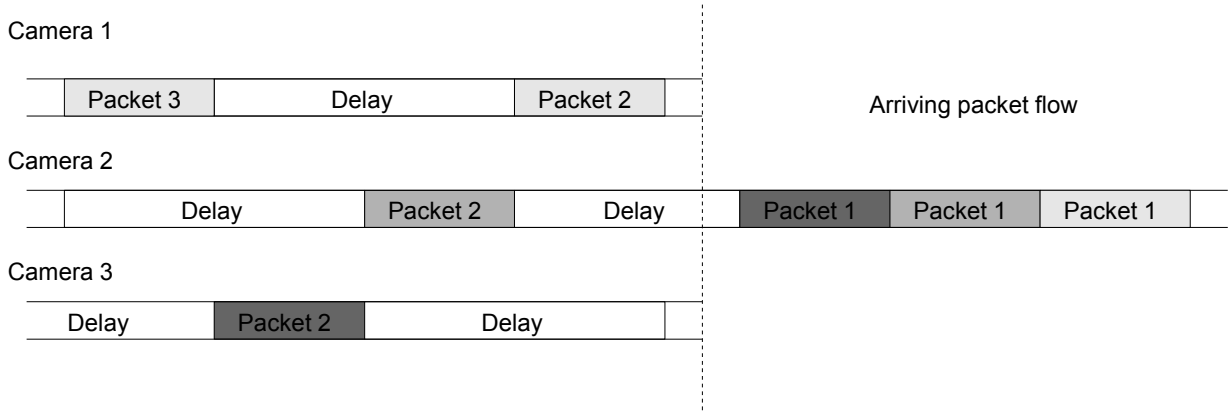


Figure 51: Packet flow example with three cameras and inter packet delay

5.6.3 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:

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

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

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

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

$$\mathrm{totalBytesInPacket} = \mathrm{GevSCPSPacketSize} + \mathrm{sizeOfEthernetHeaders}$$

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

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

Payloadsize - frame size in bytes (retrieved from camera)
effectiveBytesInPacket - number of effective bytes transferred in packet without headers

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

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

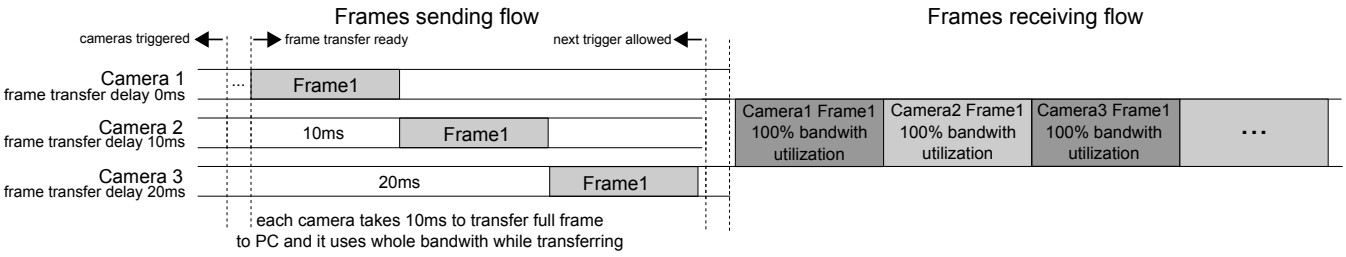


Figure 52: Example flow of frames when GevSCFTD is used

Figure 52 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.

5.6.3.1 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:

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

GevTimestampTickFrequency - indicates the number of timestamp ticks during 1 second

5.7 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.

5.7.1 Image Layout

Figure 53 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.

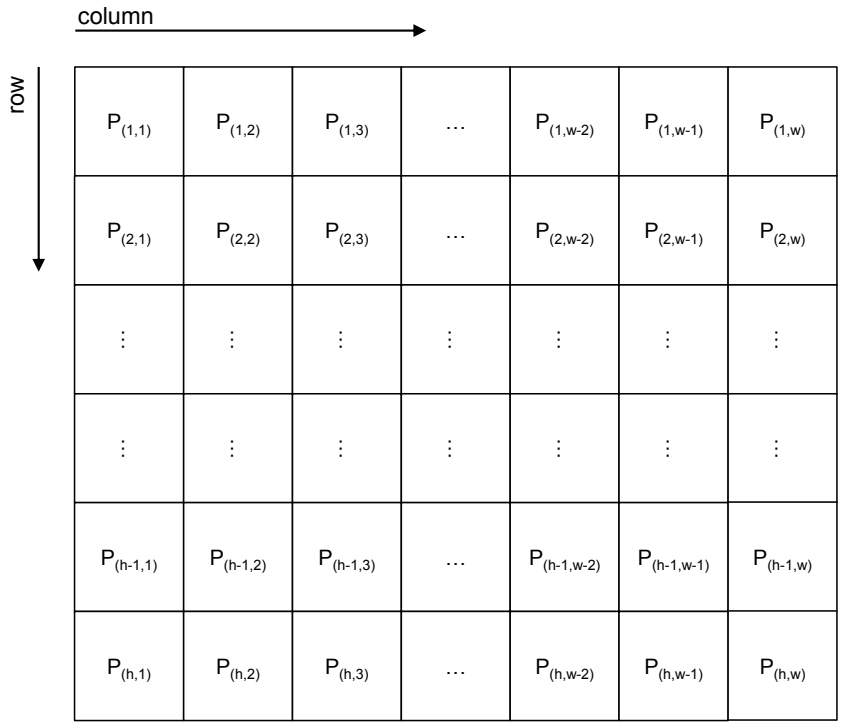


Figure 53: Image layout and transmission sequence

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

5.7.2 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 40: Specification PixelFormat Mono8

The memory layout of the image with *Mono8* pixel format is shown in Figure 54. 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.

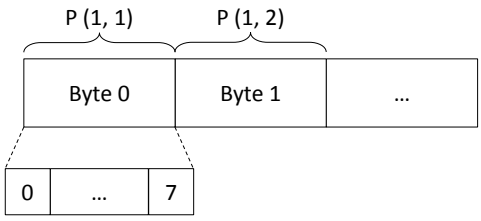


Figure 54: Image layout with pixel format Mono8

5.7.3 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 41: Specification PixelFormat Mono10Packed

The memory layout of the image with *Mono10Packed* pixel format is shown in Figure 55. 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 represent 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.

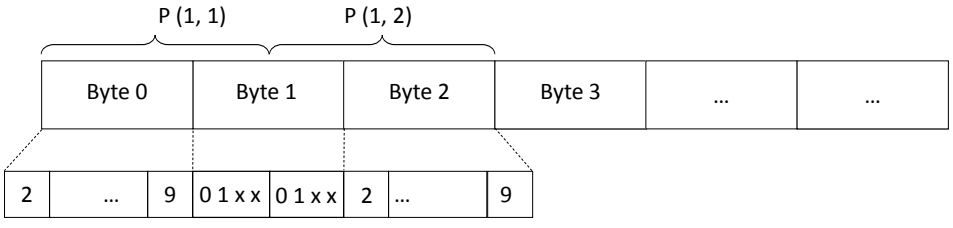


Figure 55: Image layout with pixel format Mono10Packed

5.7.4 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 42: Specification PixelFormat Mono12Packed

The memory layout of the image with *Mono12Packed* pixel format is shown in Figure 56. 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 represent 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.

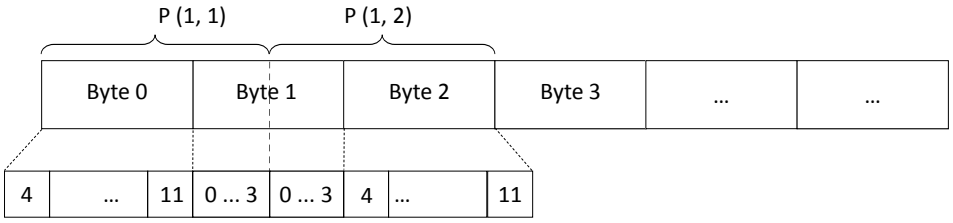


Figure 56: Image layout with pixel format Mono12Packed

5.7.5 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 43: 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 57. 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.

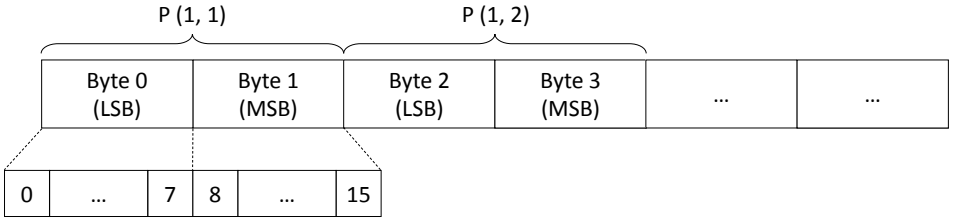


Figure 57: Image layout with pixel format Mono16

5.7.6 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 *4.2 - 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 44: Specification PixelFormat Bayer8

The memory layout of the image with this pixel formats is shown in Figure 58. 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.

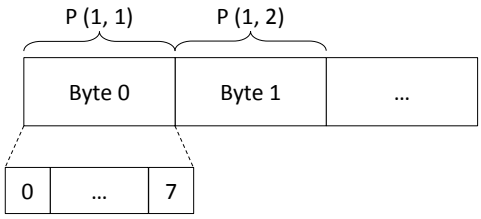


Figure 58: Image Layout with pixel format GR8/RG8/GB8/BG8

5.7.7 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 4.2 - 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 45: Specification PixelFormat Bayer16

The two bytes are arranged in little-endian order, which means the least significant byte comes first, the most significant byte second. The memory layout of the image with the *Bayer16* pixel format is shown in Figure 59. 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.

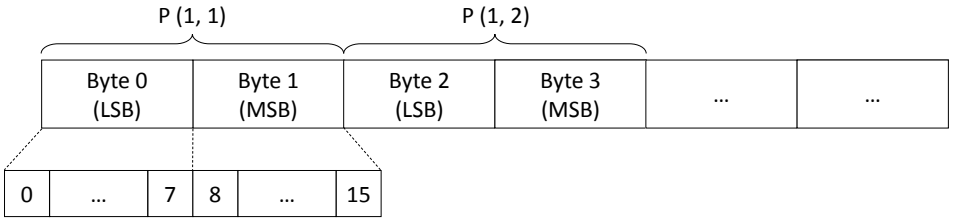


Figure 59: Image layout with pixel format GR16/RG16/GB16/BG16

6 CameraSuite SDK

The SMARTEK Vision CameraSuite Software Development Kit (SDK) provides a set of tools, guides and samples, useful for the configuration and image acquisition from GigE Vision and USB3 Vision cameras, as well as their integration into customer specific software applications. The CameraSuite SDK consists of the following components:

- **CameraSuite API** - Application Programming Interface (API) for configuration and image acquisition from GigE Vision and USB3 Vision digital cameras. Supports C, C++, Delphi, C# and VB.NET programming languages
- **Smartek GigE Vision filter Driver** - High-performance network filter driver designed to ensure optimal performance of the GigE Vision cameras
- **Smartek USB3 Vision Device Driver** - High-performance USB3 Vision device driver designed to ensure optimal performance of the USB3 Vision cameras
- **ImageProc API** - The ImageProc API extends the basic camera functionality, provided by the CameraSuite API, by color and post processing functions like e.g. debayering, gamma, look-up table (LUT) and color correction algorithms
- **Example code** - Example source code for various CameraSuite API functions available in C, C++, C++ MFC, C#, VB.NET, Delphi programming languages
- **CameraSuiteClient** - An example application utilizing the entire CameraSuite API function set in an intuitive graphical user interface

6.1 Supported Operating Systems

The SMARTEK Vision CameraSuite SDK has been created to support Microsoft Windows 7, 8, 8.1 and 10 operating systems. The installation packages for the supported operating systems are available in 32-bit(x86) and 64-bit(x64) system architecture. Table 46 contains a list of the supported operating systems and the appropriate installation package.

	Operating System	32 Bit	64 Bit	SDK Package
Microsoft	Windows 7	✓	✓	Windows Executable
	Windows 8	✓	✓	
	Windows 8.1	✓	✓	
	Windows 10	✓	✓	

Table 46: Supported operating systems and installation packages

6.2 Installation of the CameraSuite SDK

The installation packages for the latest CameraSuite SDK can be downloaded from the SMARTEK Vision webpage:

http://www.SMARTEK.vision/media-center

After downloading and executing the installation package, the user can choose which part of the SDK to install, shown in Figure 60.

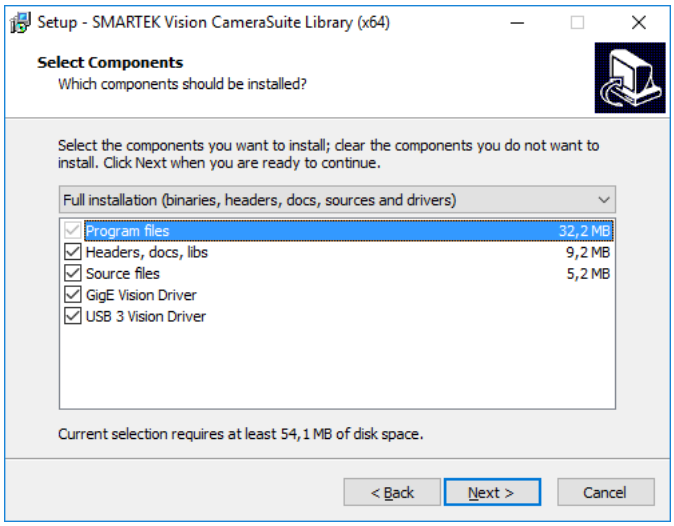


Figure 60: CameraSuite SDK Setup

The installation assistent will guide the user through the installation procedure.

During installation of the SDK the network connections may temporarily become unavailable. Please ensure all applications are closed that are accessing the network connection.

6.3 Manual Driver Installation

6.3.1 Installation GigE Vision Filter Driver

The Smartek GigE Vision Filter Driver can be installed manually and independently of the CameraSuite SDK by executing the GigE Vision Driver Installer. The corresponding executable is installed with the CameraSuite SDK. To execute it, navigate as administrator with the Command Prompt or Windows PowerShell into the CameraSuite SDK installation directory.

The CameraSuite SDK is located by default at the following location:

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

The GigEVision Driver Installer can be found in a subfolder within the installation directory:

```
..\drivers\FilterDriver\GigEVDrvInstall.exe
```

There are several flags that control the installer execution:

- /**install** - Installs the Smartek GigE Vision Filter Driver. it can be used with optional flags like /v or /hide.
- /**uninstall** - Uninstalls the Smartek GigE Vision Filter Driver. it can be used with optional flags like /v or /hide.
- /**v** - Optional flag that runs the script with additional information.
- /**hide** - Optional flag that runs the script silently.

Installation or uninstallation for Smartek GigE Vision filter driver won't succeed if there is a instance of CameraSuite SDK running in the background!

6.3.2 Installation USB3 Vision Device Driver

If the Smartek USB3 Vision Device Drivers needs to be installed manually, it can be installed independently of the CameraSuite SDK by executing Driver Package Installer located in the SDK's installation directory after successfully installing the CameraSuite SDK or through Microsoft Windows Device Manager.

The Device Drivers can be found in a subfolder of the installation directory. By default the CameraSuite SDK is installed into the following folder:

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

The driver specific files and the Driver Packet Installer (dpinst.exe) are located in the following subfolder of the installation directory:

```
..\drivers\UsbDriver\..
```

6.4 Unattended SDK Installation (Microsoft Windows)

In cases where the CameraSuite SDK needs to be included into 3rd-party installation routines, 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 or Windows PowerShell as Administrator, with the following flags (in one line):

```
1 SMARTEKvision_CameraSuite_Vxxxx.exe /VERYSILENT /SUPPRESSMSGBOXES /NORESTART /TYPE="minimal"
2
```

There are three types of installations that can be performed:

1. `/TYPE="minimal"` - Minimal installation (binaries only)
2. `/TYPE="compact"` - Compact installation (binaries, examples, docs and drivers)
3. `/TYPE="full"` - Full installation (binaries, examples, docs, sources and drivers)

6.5 CameraSuiteClient

The CameraSuiteClient is a Qt-based open-source application installed along with the CameraSuite SDK. 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 or USB3 Vision complaint cameras.

After the installation of the CameraSuite SDK the CameraSuiteClient 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\CameraSuite\bin\
```

The source code is located at:

```
C:\Program Files\SMARTEKvision\CameraSuite\CameraSuiteClient\..
```

6.5.1 Graphical User Interface (GUI)

Figure 61 shows the default GUI elements of the CameraSuiteClient.

- **Menu Bar** - always on top of the application. It provides access to all functionalities available on the toolbar. Also main dialogs within the CameraSuiteClient 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 and USB3 Vision compliant devices and their 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 parameterization 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.

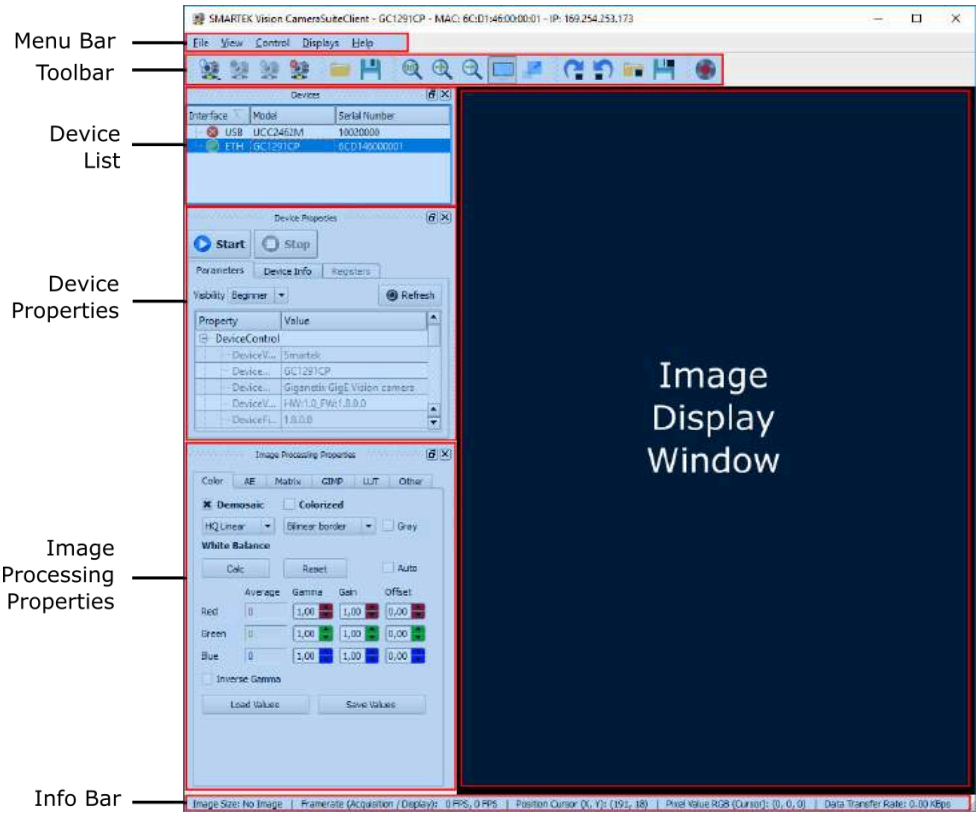


Figure 61: CameraSuiteClient Graphical User Interface (GUI)

Refer to Figure 62 for a full description of all GUI elements on the toolbar.

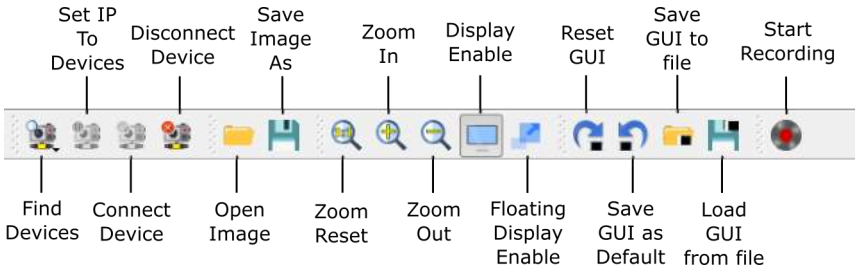


Figure 62: Toolbar description

Due to the possibility that all the dialogs within the CameraSuiteClient 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 CameraSuiteClient installed on other machines. The GUI save and reset features are accessed through the menu bar or the toolbar, as shown in Figure 62.

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

6.5.2 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 CameraSuiteClient.

6.5.2.1 Device Enumeration

After the CameraSuiteClient is started, the user can click on *Find Devices* button to start the sequence for finding devices, shown in Figure 63. All found devices are listed in the *Device List* dialog.

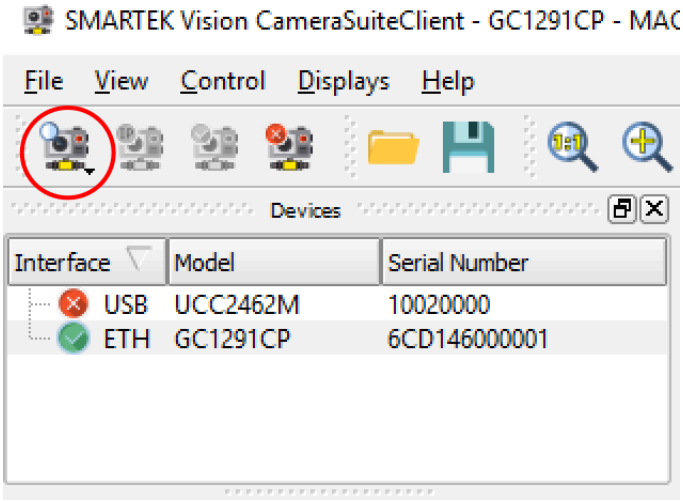


Figure 63: Find Devices icon

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 multiple reasons:

For GigE Vision compliant devices:

1. The IP-address and subnet mask of the network interface card or camera are invalid
2. The link speed of the used network interface is providing less than 1000 Mbit/s
3. The device uses the GigE Vision filter driver from another manufacturer

For USB3 Vision compliant devices:

1. No USB3 Vision device driver is installed correctly on the device
2. The device uses the USB3 Vision device driver from another manufacturer

For a quick start (if you are using GigE Vision cameras) it is recommended to only use Gigabit Ethernet NICs, configured according to Table 48 shown in chapter *7.1.2 - 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.


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 *3.1 - Ethernet Interface*. Make sure everything is plugged properly and that the firewall settings are not blocking the connection to camera or CameraSuiteClient.

6.5.2.2 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 64.

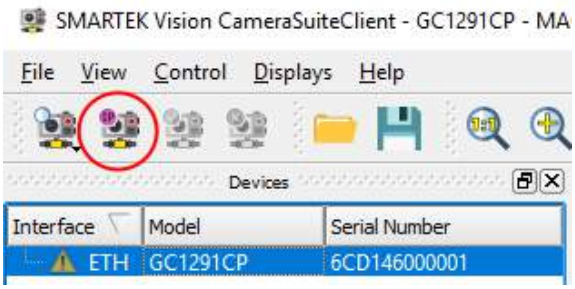
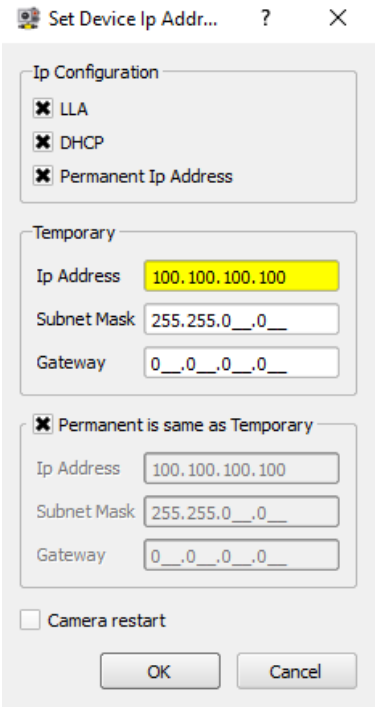
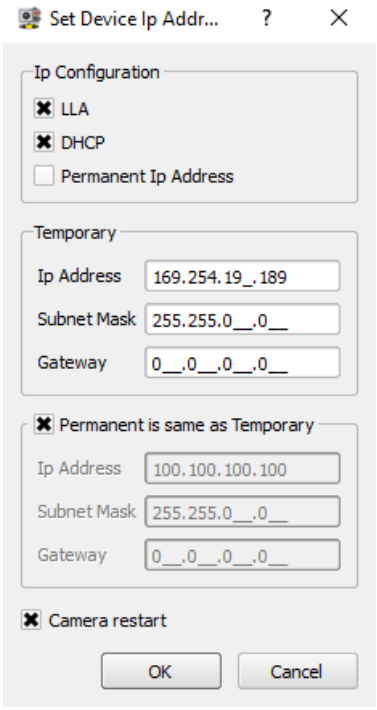


Figure 64: Set Ip To Device icon

A new window will open showing an access mask to the IP address, subnet mask and gateway configuration of the chosen camera. If one of the fields in *Temporary* is not valid (i.e. the IP Address is within the wrong subnet or already taken), the color of that field is going to change to yellow, shown in Figure 65a. With the *Permanent is same as Temporary* checkbox unchecked, it is possible to set a temporary IP address that works for the current subnet value, but is after a camera restart changed to permanent.



(a) Temporary IP Address is in wrong subnet



(b) Temporary IP Address is in right subnet

Figure 65: Set Device IP Address dialog

After setting a valid IP address configuration to the camera, the warning symbol next to the camera model name will change like shown in Figure 66, the *Connect Device* icon can now be used to connect to the selected camera.

Figure 66: Connect Device icon

6.5.2.3 Device Properties

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

General information about the camera selected from the list of discovered devices is 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 67 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.

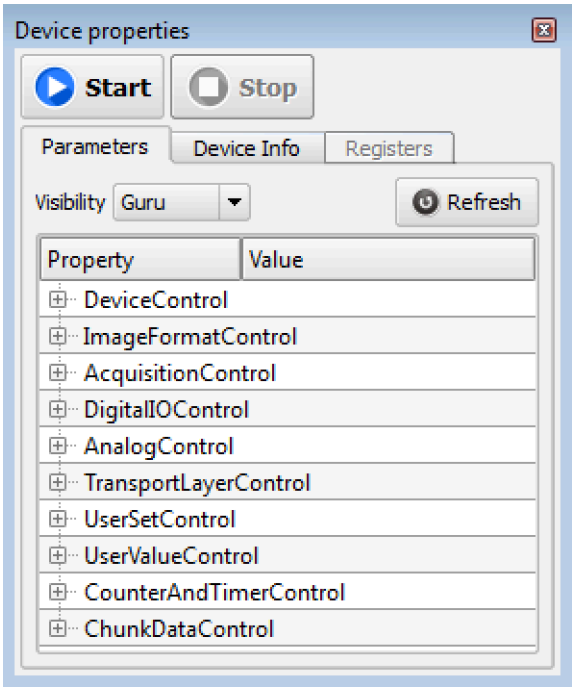


Figure 67: Device Properties - Parameters tab

According to the current settings of the camera, the acquisition can be started by pressing the *Start* button shown in Figure 68. 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*.

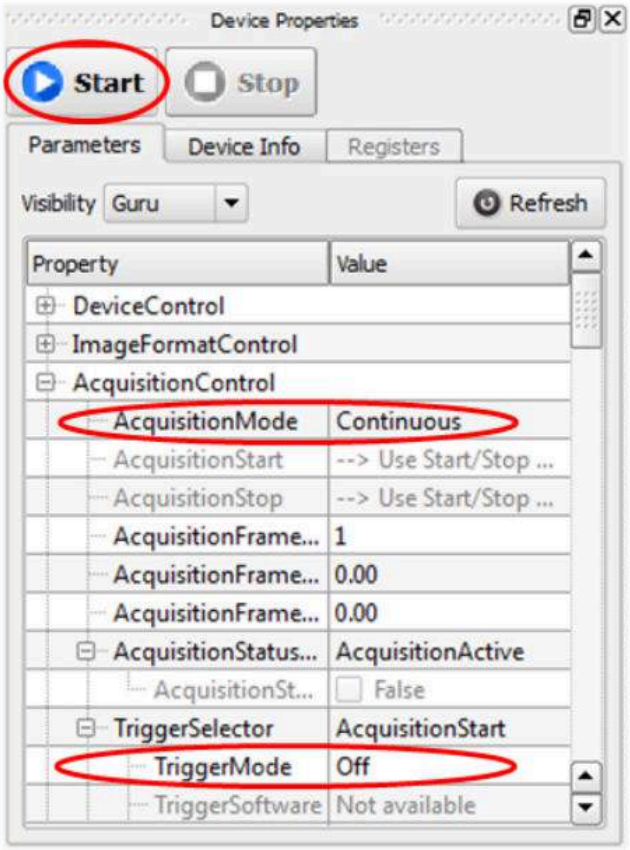


Figure 68: Device Properties - Starting a continuous stream

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.

6.5.2.4 Multiple Devices, Multiple Views

The CameraSuiteClient supports the operation of multiple devices that are connected and detected in parallel. The video stream of each device can be displayed in separated floating windows or grouped together in a *Preview* dialog. Figure 69 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.

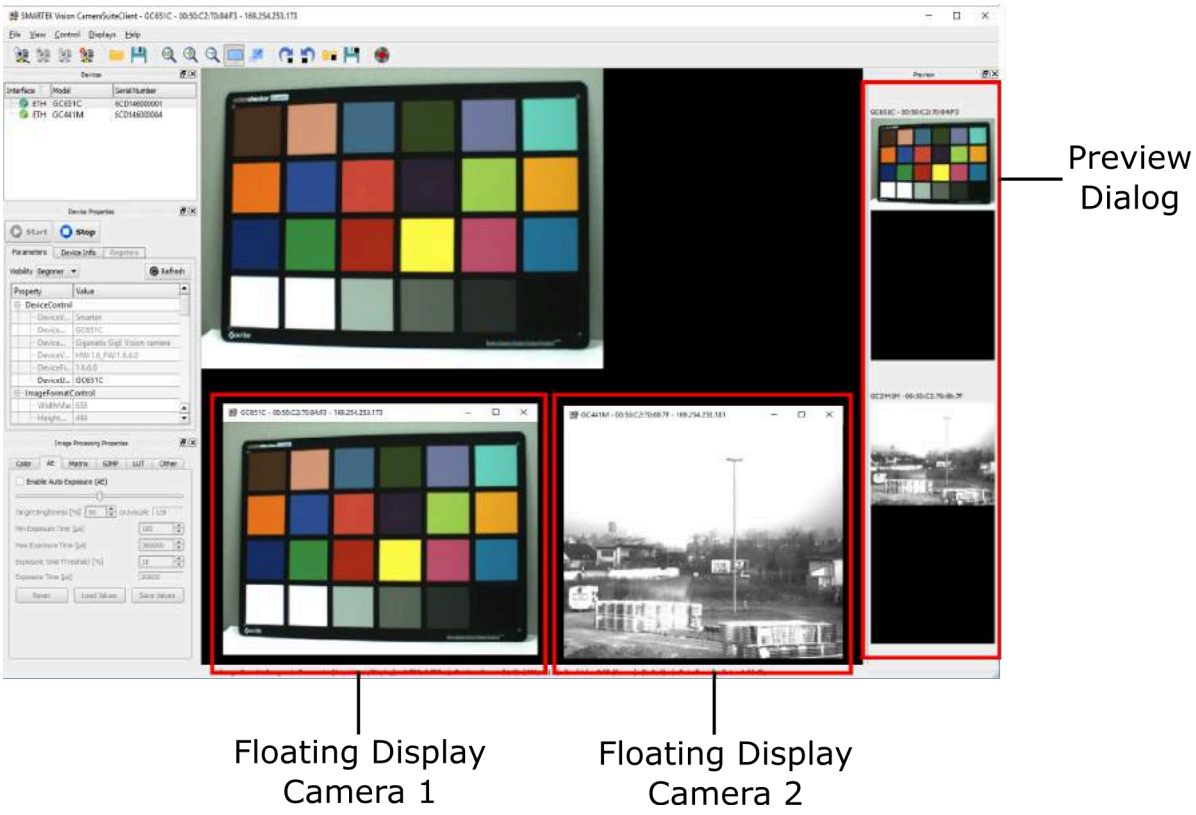


Figure 69: Preview Dialog and Floating Displays for multiple cameras

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.

6.5.2.5 Image Processing

Image processing algorithms provided by the ImageProc API can be accessed within the *Image Processing* dialog, shown in Figure 70. It enables the user to apply and parameterize the available image preprocessing functions.

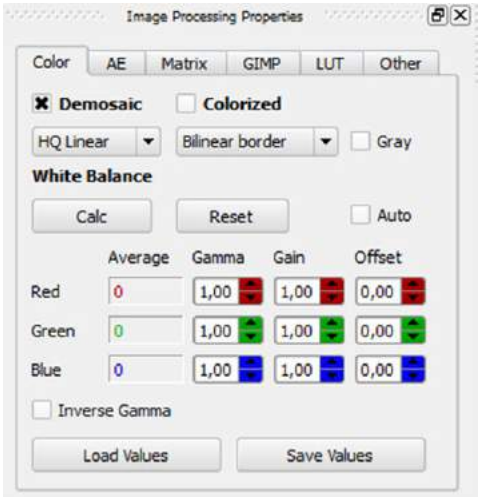


Figure 70: Image processing properties - Color tab

Table 47 shows an overview of the tabs and functions each tab includes. A deep description of all listed image functions can be found in chapter 8 - Image Processing in *CameraSuite SDK*.

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 47: CameraSuiteClient - Image processing functions

6.5.3 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 71, allows the modification of all available API parameters and gives access to the packet resend settings and statistics of the current driver session. Further it enables the user to rise or decrease the image buffer within the camera driver, which 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 to 75% of physical RAM), the number of images in the image buffer must be calculated correctly.

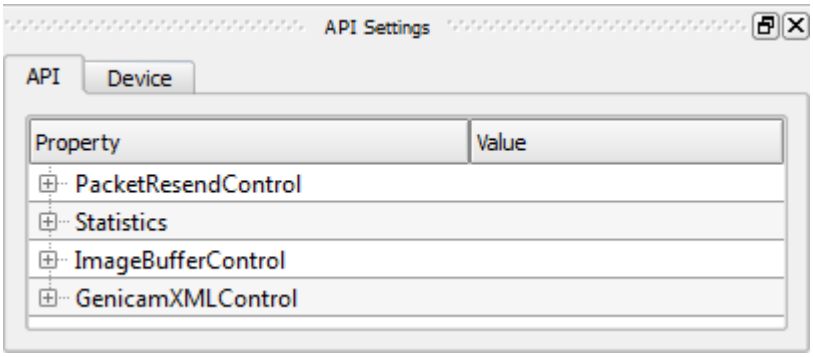


Figure 71: API Settings

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

6.5.4 Chunk Data Control

To access chunk data control in CameraSuiteClient, the property visibility must be set to Expert or Guru. Controls are located at bottom of the Device Properties window. Figure 72 shows chunk data control.

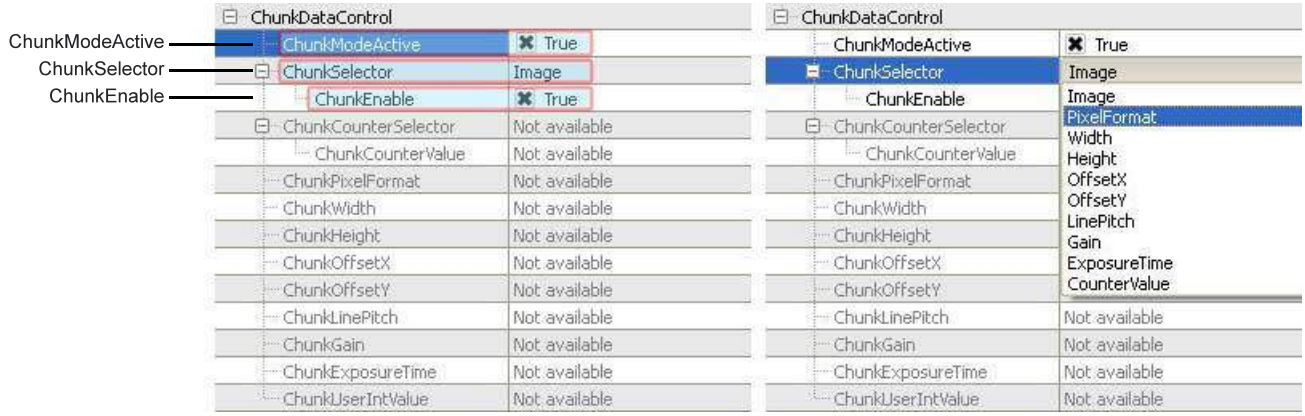


Figure 72: Chunk Data Control

- **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 73 shows chunk data values tab.

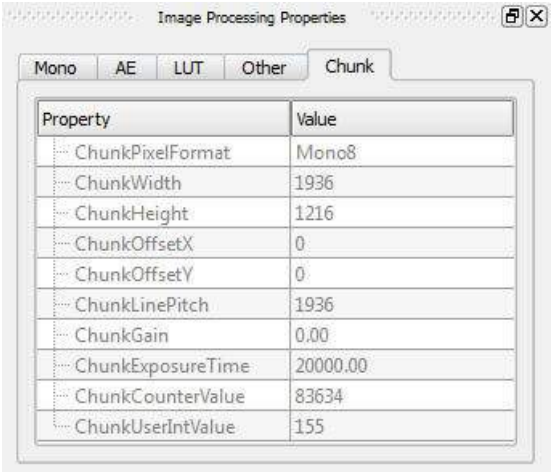


Figure 73: Chunk Data Values

Setting Chunk Data values

UserIntValue can be set up through properties window shown in Figure 74. The value is included in Data Payload if UserIntValue chunk is enabled.

Figure 74: Chunk Data Values

Figure 75 shows CounterAndTimerControl.

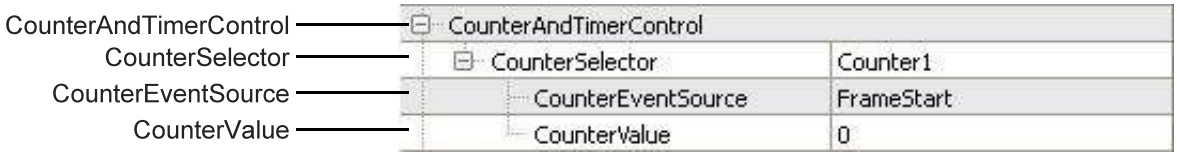


Figure 75: Chunk Data Values

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

6.5.5 Log Dialog

The Log dialog contains logging information from SDK and cameras, shown in Figure 76.

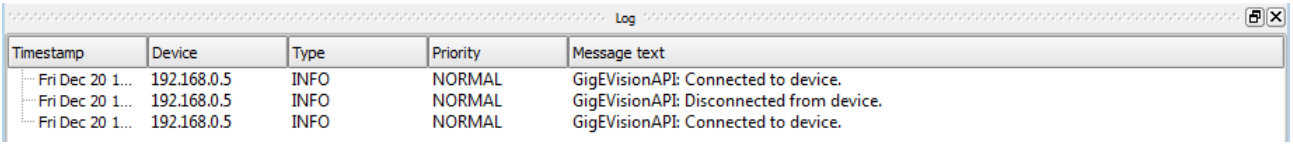


Figure 76: Log dialog with API logging

6.5.6 Firmware Update

The *CameraSuiteClient* contains a module to update the firmware of a twentynine camera. To start the firmware update process, connect with the CameraSuiteClient to the target camera and open the *Firmware Update* dialog via the *Control* category in the menu bar.

In the Firmware Update Dialog, shown in Figure 77, follow the following steps:

1. **Browse for Firmware Updates**
   Search and select a firmware binary 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.
2. **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.
3. **Flash Firmware**
   To upload and flash the firmware to the target camera, press the *Upload new firmware to device* button.
4. **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.

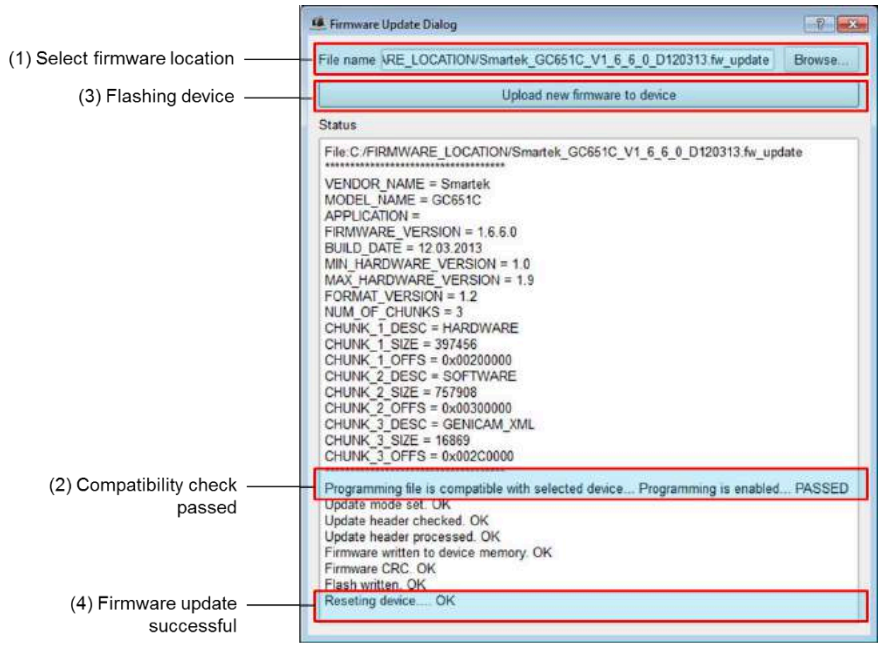


Figure 77: Firmware update dialog after the update is successfully executed

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

7 GigE Vision / USB3 Vision Specific Notes

7.1 GigE Vision

7.1.1 Choosing 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 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.

7.1.2 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 48 shows a good configuration for a first start with a single NIC, but does however not represent a setup suited for all application and network environments.

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

Table 48: Basic IP configuration of the PC

Note

To use 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.

7.1.2.1 IP Setup in Microsoft Windows

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

1. Execute "ncpa.cpl", i.e. 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 78, usually provided by routers or dedicated servers. A fixed IP address and Subnet mask can be entered like shown in Figure 78 right, after choosing *Use the following IP address*.

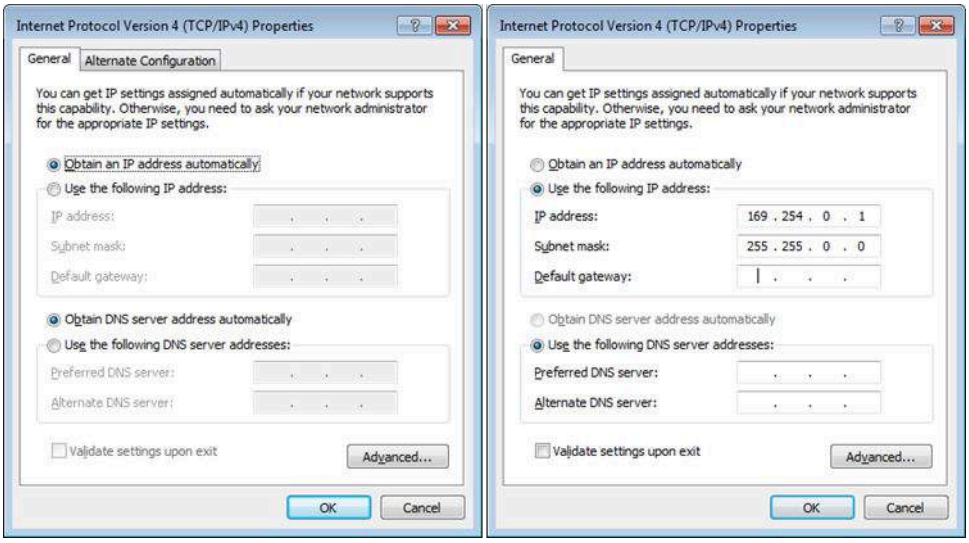


Figure 78: Internet Protocol Version 4 (TCP/IPv4) properties

7.1.3 Network Interface Optimization

To reach the optimal performance with the twentynine 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.

7.1.3.1 Jumbo Frames - Network Interface Cards

For Network Interface Cards (NIC) 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 the following way:

1. Execute "ncpa.cpl", i.e. 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 79.

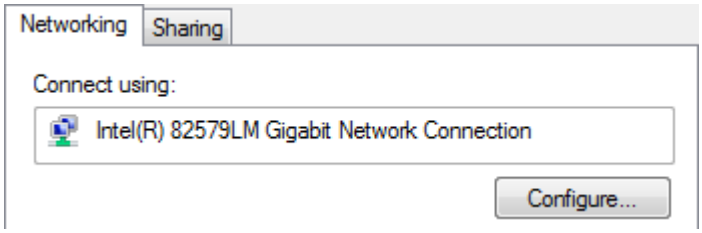


Figure 79: Access to network interface card driver settings

4. The settings of the driver can be accessed by choosing the *Advanced* tab of the opened window. As shown in Figure 80, raise the value of the *Jumbo Packet* property to its maximum value.

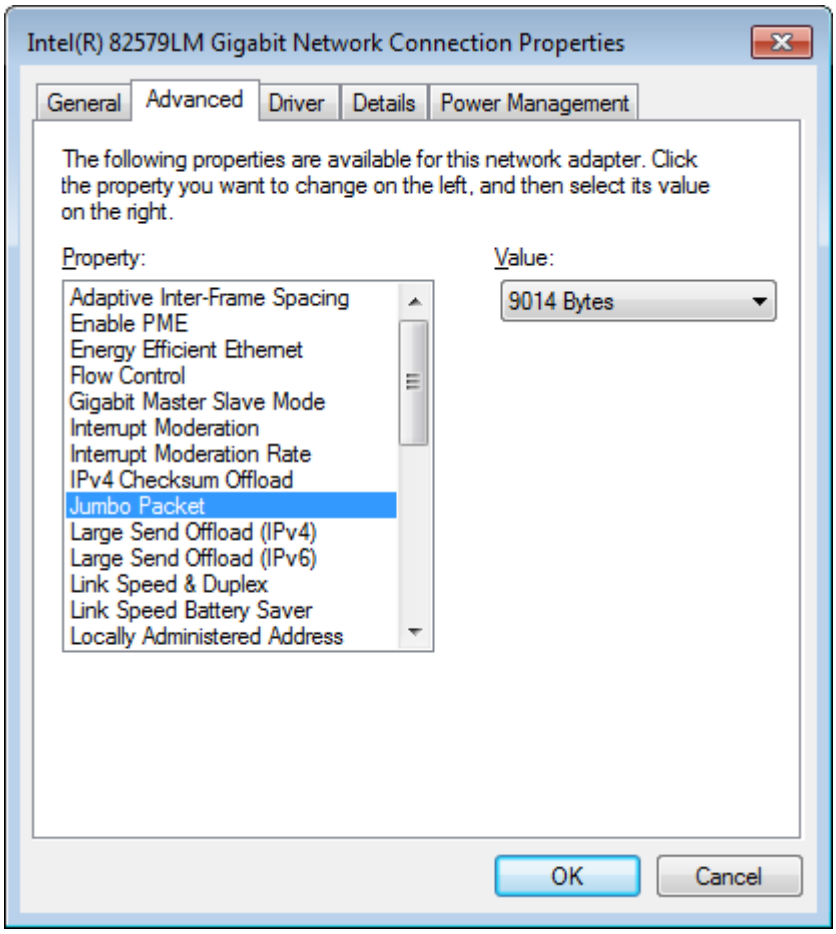


Figure 80: Network interface card - advanced driver settings - Jumbo Packets

7.1.3.2 Jumbo Frames - 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.

7.1.3.3 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 the following way:

1. Execute "ncpa.cpl", i.e. 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 79.
4. The settings of the driver can be accessed by choosing the *Advanced* tab of the opened window. As shown in Figure 81, raise the value of the *Receive Buffers* property to its maximum value.

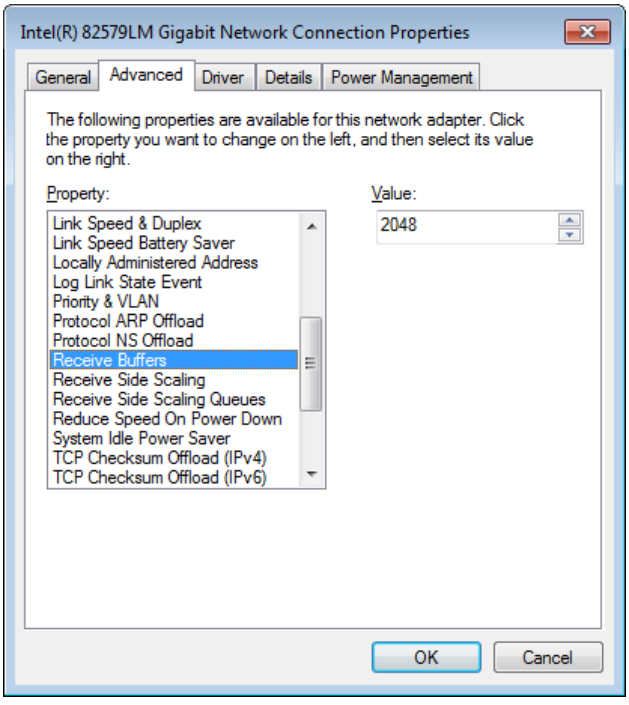


Figure 81: Network interface card - advanced driver settings - Receive Buffers

7.1.3.4 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 82.

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

1. Execute "ncpa.cpl", i.e. 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 79.
4. The settings of the driver can be accessed by choosing the *Advanced* tab of the opened window. As shown in Figure 82, set the value of the *Interrupt Moderation Rate* property to *Off* or *Disabled*.

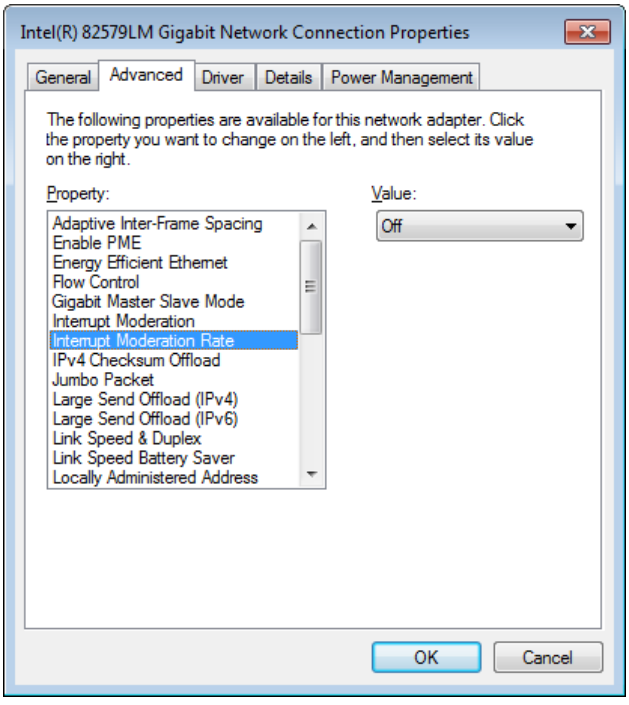


Figure 82: Network interface card - advanced driver settings - Interrupt Moderation Rate

7.1.3.5 Disable the Flow Control

To further optimize the latency of the data transmission, shown in *7.1.3.4 - 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 the following way:

1. Execute "ncpa.cpl", i.e. 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 79.
4. The settings of the driver can be accessed by choosing the *Advanced* tab of the opened window. As shown in Figure 83 , set the value of the *Interrupt Moderation Rate* property to *Disabled*.

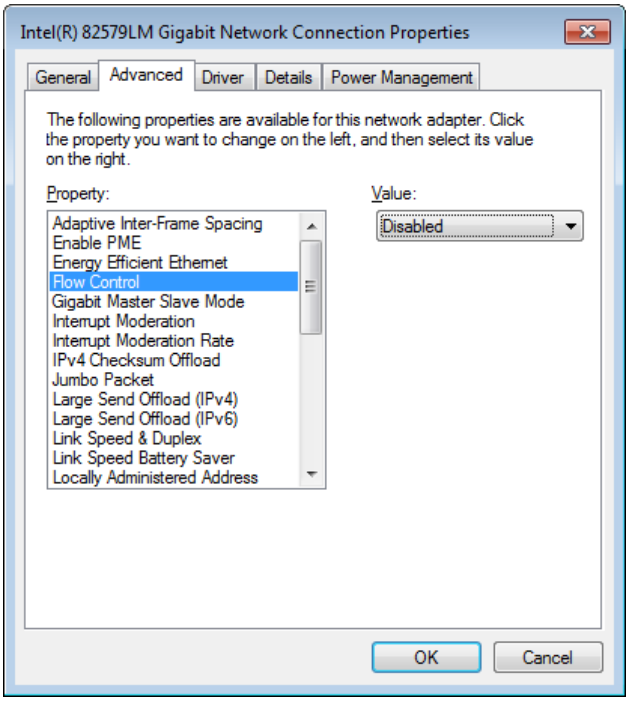


Figure 83: Network interface card - advanced driver settings - Flow Control

7.2 USB3 Vision

7.2.1 Choosing the right USB3.0 Host Controller

To set up a stable high performance USB 3.0 system, all the system components must be carefully chosen to sustain large amount of data generated by the USB3 Vision twentynine camera. The most important component is the USB3.0 Host Controller as some of the controllers may degrade the performance drastically.

Recommended USB3.0 Host Controllers:

- PC Motherboards with Intel Haswell or newer chipsets
- PCI Express (PCI-E) host adapter cards with:
  - Fresco Logic FL1100 USB3.0 host controller chipset
  - Renesas uPD720201 USB3.0 host controller chipsets

7.2.2 USB 3.0 Cabling Recommendation

To connect the camera to a PC, a USB3.0 A to micro-B cable is required. As USB3.0 is a very compact high-speed interface, it is highly recommended to only use industrial grade cabling assembled with high-grade connectors providing appropiate locking to the camera port. The ratio between high-frequency characteristics and cooper wire gauge results into a maximum possible cable length. To avoid EMI, it is recommendet to use only shielded cables, a close installation to high frequency electromagnetic fields should be avoided.


Note

Bad cabling, connectors and/or shielding can lead to decreased performance (e.g. framerates) up to connection interrupts. It is thus highly recommended to purchase the right industrial cabling from our local SMARTEK Vision distribution partner to prevent issue in performance.

8 Image Processing in CameraSuite SDK

The *ImageProcAPI* provided by the *CameraSuite SDK* 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 *CameraSuite* API are supported by the *ImageProcAPI* as well, namely C/C++, Delphi, C# and VB .NET.

Table 49 lists all image processing algorithms implemented in the *ImageProcAPI* including the supported input image type.

	Supported image input type		
Image statistics	**Monochrome**	**Raw Bayer**	**RGB**
Histogram	✓	✓	✓
Average luminance	✓	✓	✓
Image processing algorithms			
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	✓	✓	✓
Image Processing pipeline			
Color pipeline	✓	✓	

Table 49: 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 *CameraSuite SDK* installation directory. For a more detailed description on the parameters of each algorithm or on how to apply them, please refer to the *CameraSuite* API help located at the doc folder of the *CameraSuite SDK* installation directory.

8.1 Image Statistics

8.1.1 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* x *image_height*, describing the total numbers of pixels for each pixel value found in the image.

Figure 84 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.

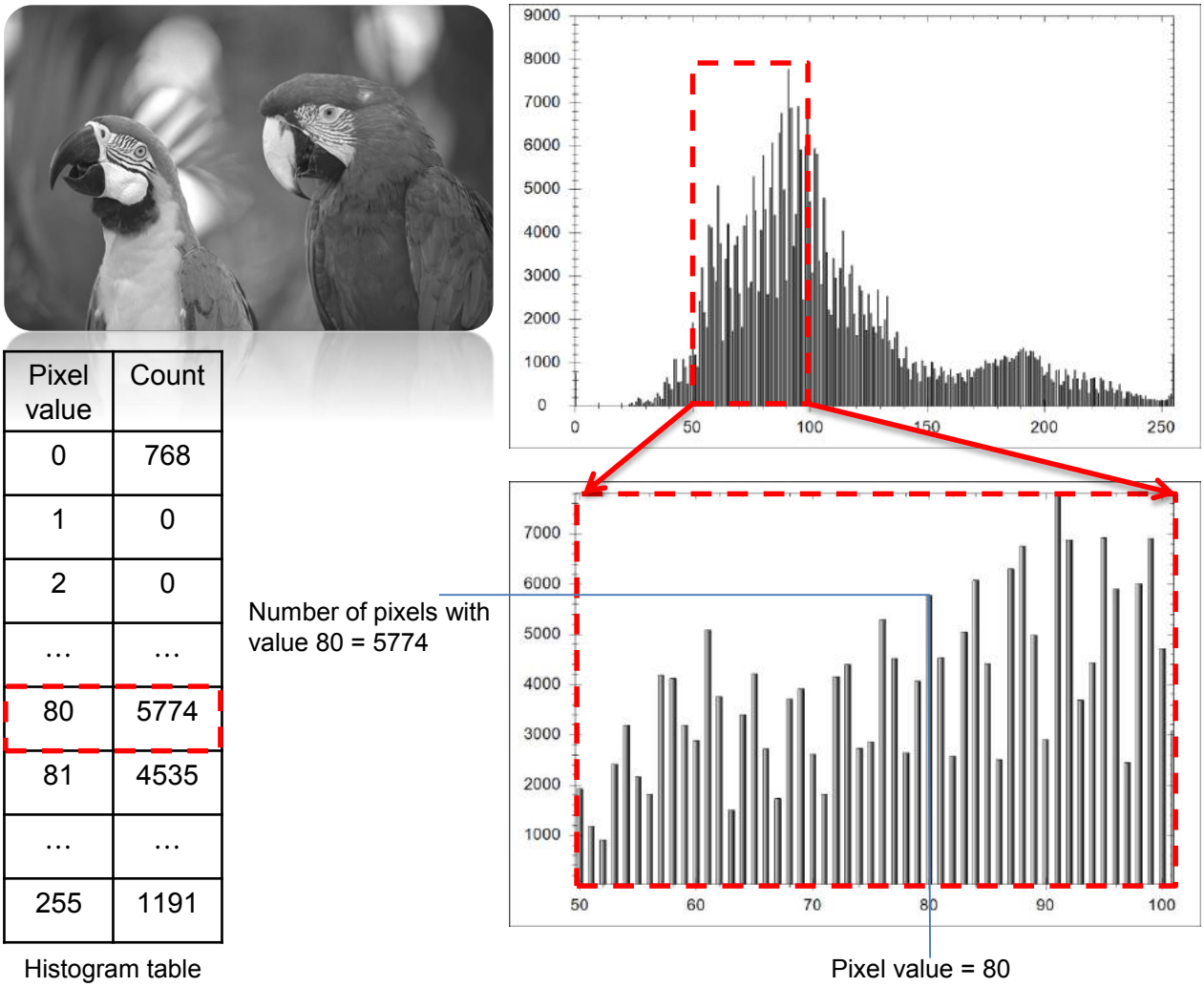
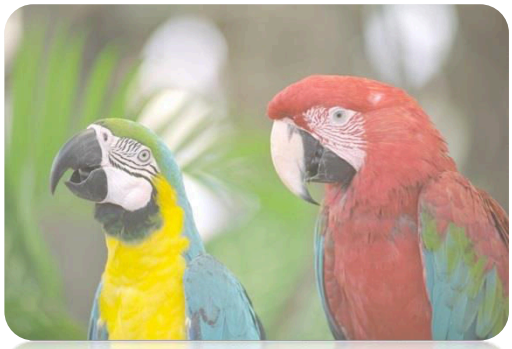


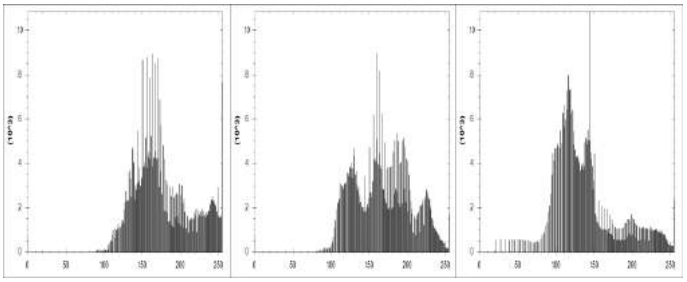
Figure 84: Source image with histogram graph and table

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 85. 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 85 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:

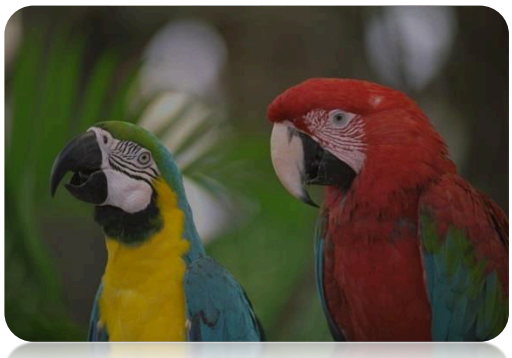
- In the first row the population of the pixel values in each histogram is shifted to the right of the center (brighter region); it indicates an overexposed image.
- In the second row the three histograms are shifted to the left (darker region); it indicates an underexposed image.
- In the third and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



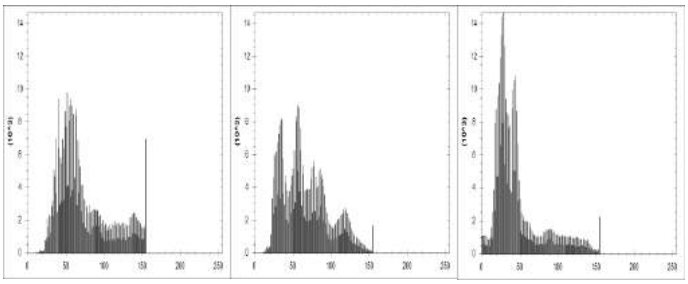
Overexposed Image



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

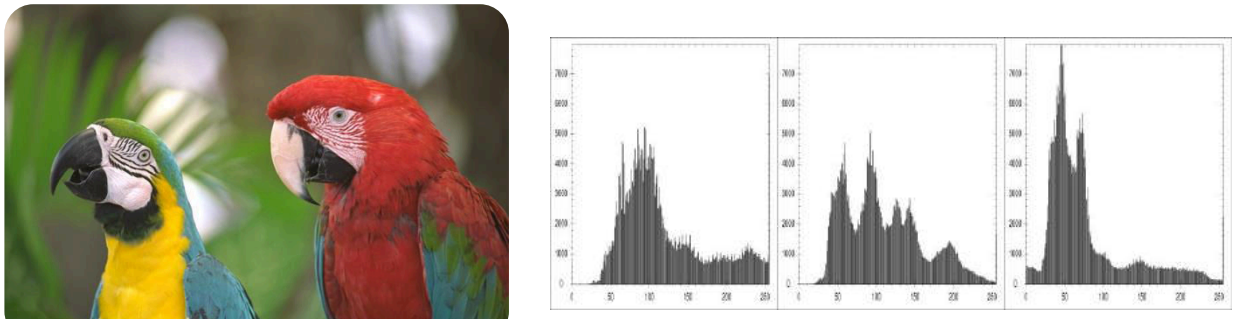


Figure 85: Example of using histogram to determine optimal image exposure

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 CameraSuite SDK

In the *CameraSuite SDK* 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 50. For a detailed description on how to instantiate this feature in user applications please refer to the *CameraSuite API Help* located in the doc folder of the *CameraSuite SDK* installation directory.

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

Table 50: Histogram - supported bit depth and image types

Histogram in the CameraSuiteClient

In the *CameraSuiteClient* application the histogram feature can be enabled by the menu bar entry Control ⇒ Histogram, shown in Figure 86.

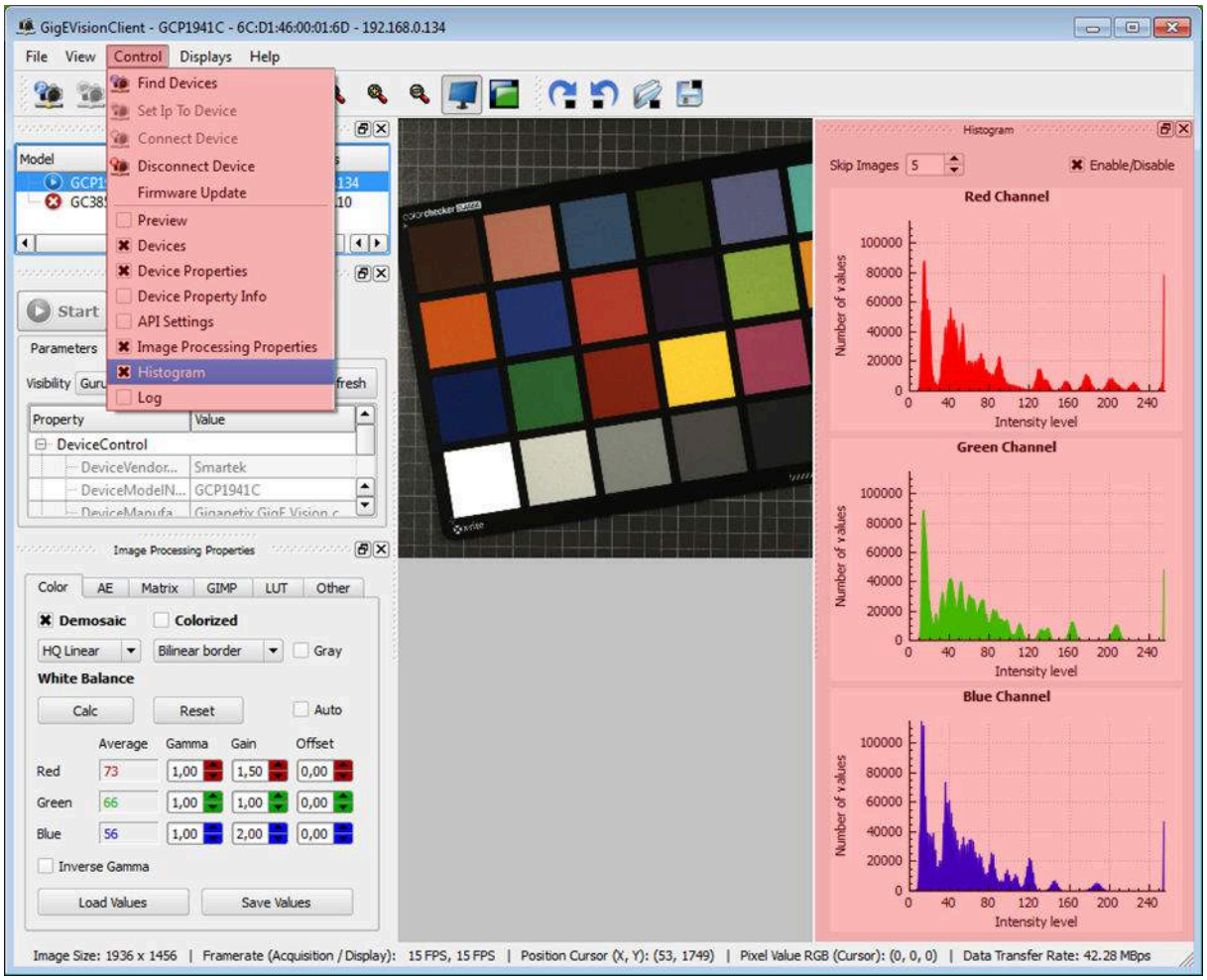


Figure 86: Histogram feature in CameraSuiteClient

- **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.

8.1.2 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:

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

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 CameraSuite SDK

In the *CameraSuite SDK* 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 51. For a detailed description on how to use the average value calculation feature please refer to the *CameraSuite API Help* located in the doc folder of the *CameraSuite SDK* installation directory.

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

Table 51: Average luminance calculation - supported bit depth and image types

Average Luminance in the CameraSuiteClient

Calculated average value(s) of the image channel(s) can be found in the *Image Processing Properties* under *Color / Mono*, shown in Figure 87. If not visible, it can be enabled by the menu bar entry *Control* ⇒ *Image Processing Properties*.

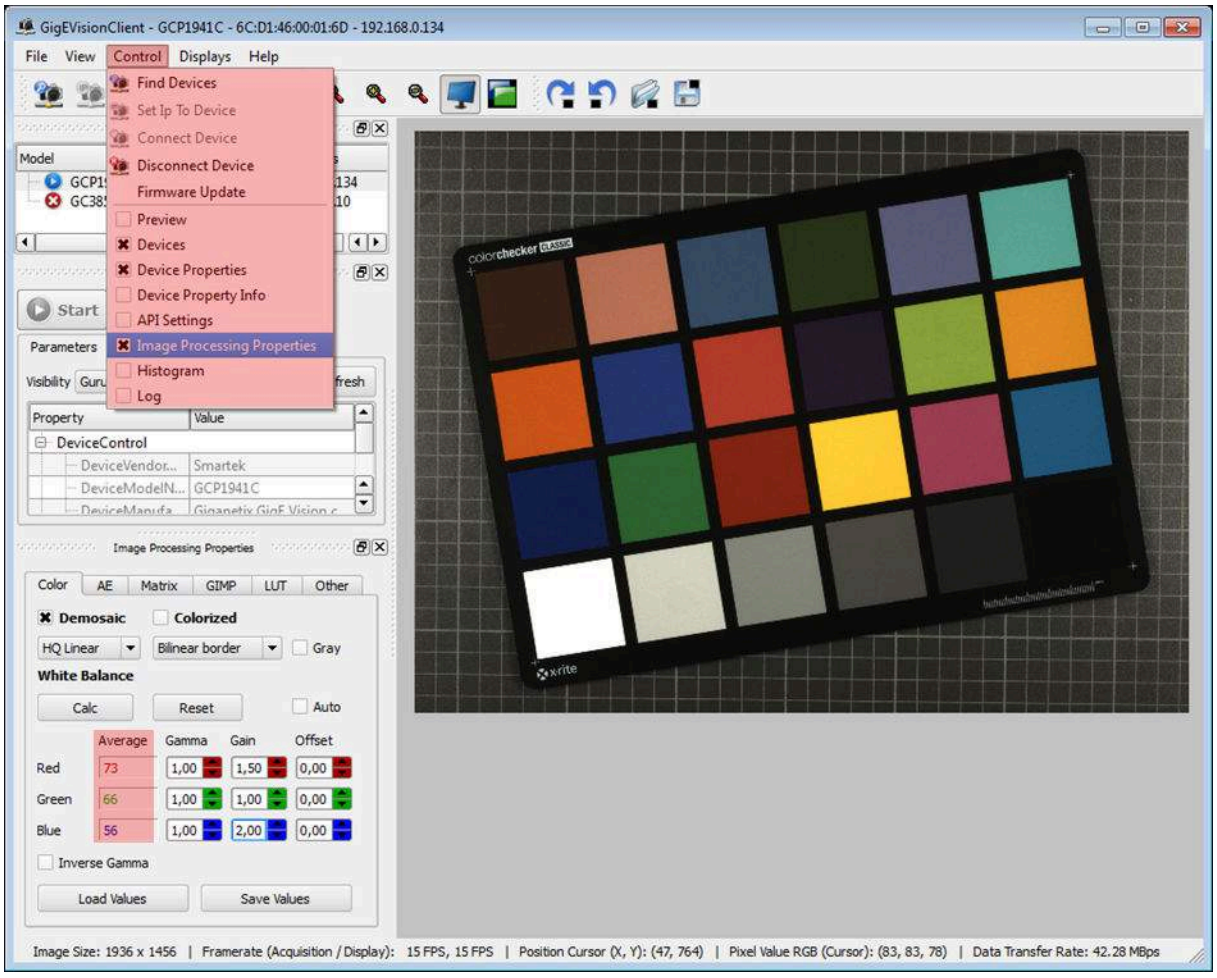


Figure 87: Average value calculation in CameraSuiteClient for White Balancing algorithm

8.2 Image Processing Algorithms

8.2.1 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 88 shows an example of an 8 bit look-up table.

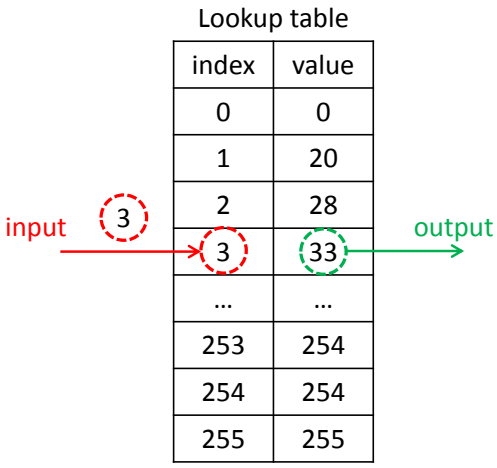


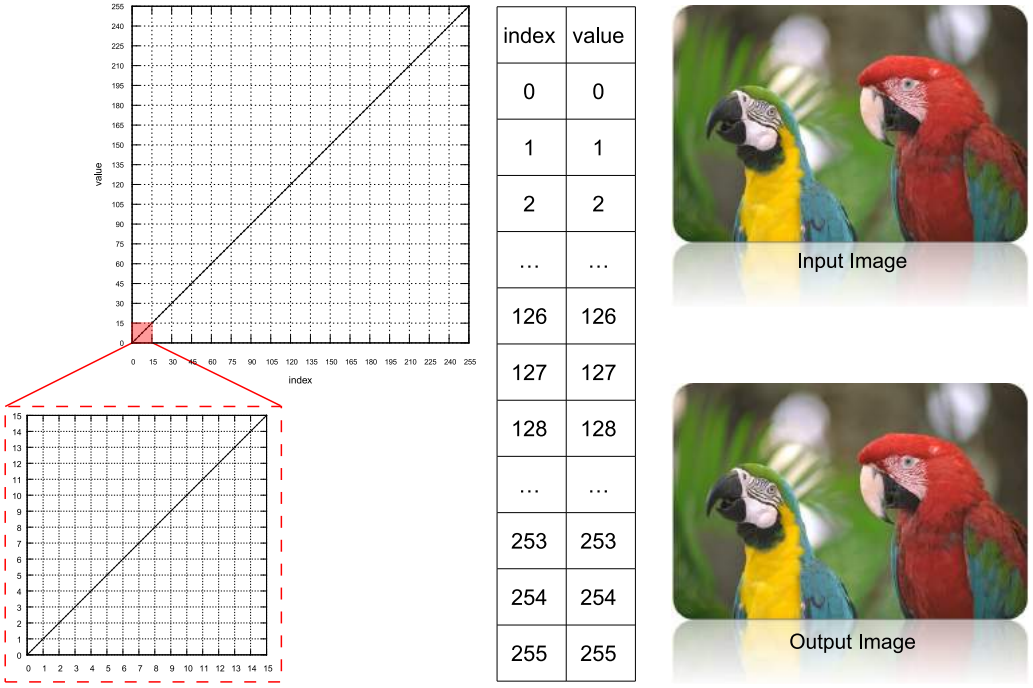
Figure 88: 8 bit look-up table

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.

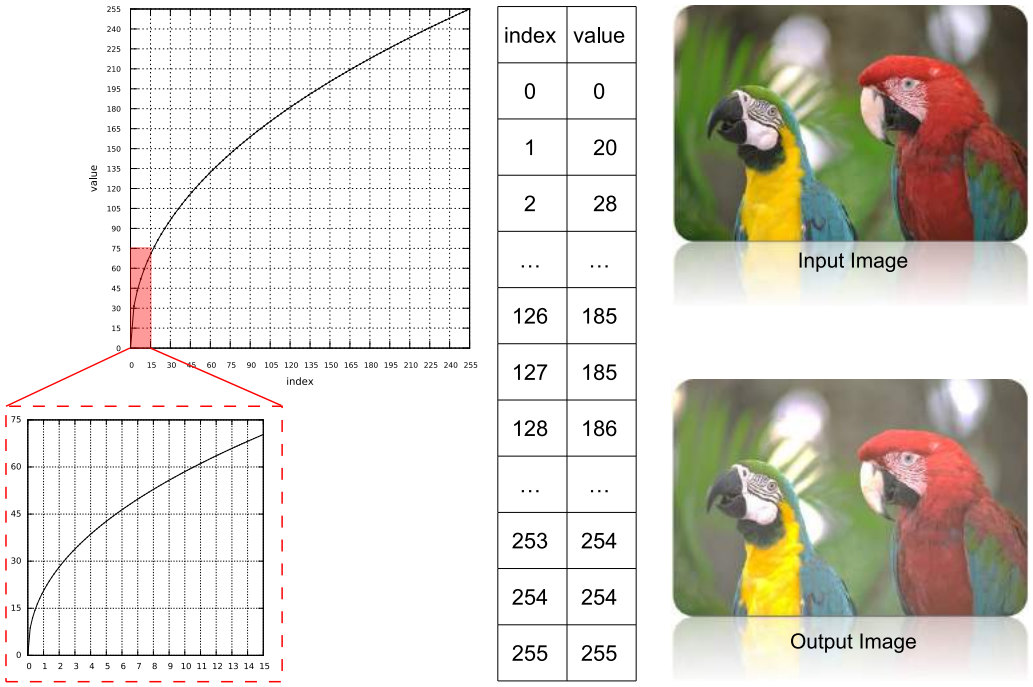
The first example shows a look-up table where each output pixel value is mapped to its exactly corresponding input pixel value. The look-up table is a linear function (f(x) = x) and its graph is a 45° straight line, shown in Figure 89. Because of the one-to-one value mapping the output image is identical to the input image.



index	value
0	0
1	1
2	2
…	…
126	126
127	127
128	128
…	…
253	253
254	254
255	255

Figure 89: Linear transformation using 8 bit look-up table

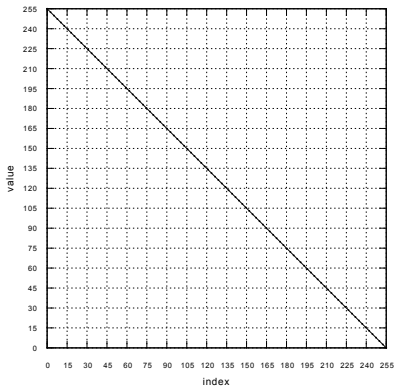
The second example demonstrates a variant of gamma correction using a look up table. By reference to the look-up table and its corresponding graph, in Figure 90, it is visible that a non-linear transformation is applied to the input pixel values.



index	value
0	0
1	20
2	28
…	…
126	185
127	185
128	186
…	…
253	254
254	254
255	255

Figure 90: Gamma correction using look-up table

The third example, illustrated in Figure 91, 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 $\text{Value}_{\text{out}} = 2^8 - \text{Value}_{\text{in}}$.



index	value
0	255
1	254
2	253
…	…
126	129
127	128
128	127
…	…
253	2
254	1
255	0

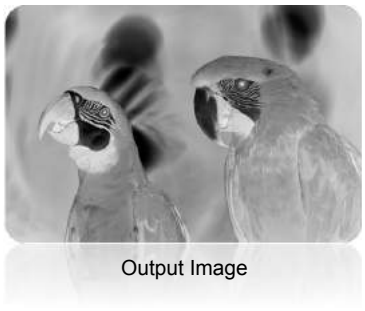
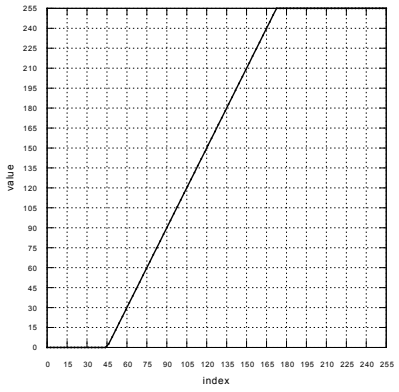


Figure 91: Inverting a monochrome image using look-up table

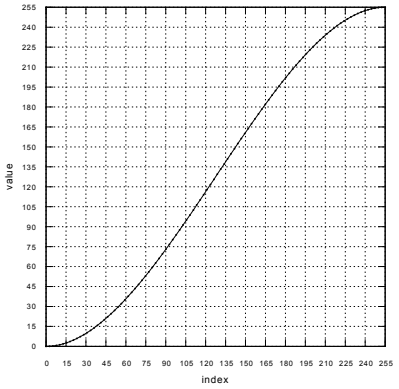
The last example demonstrates two implementations of contrast enhancement, using a look-up table applied to an 8-bit per channel color image. In Figure 92 the first 45 pixel values have been set to 0 and pixel values in range from 173 to 255 have been set to 255.



index	value
0	0
1	0
2	0
…	…
126	162
127	164
128	166
…	…
253	255
254	255
255	255

Figure 92: Enhancing contrast of an image using look-up table

Figure 93 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.



index	value
0	0
1	0
2	0
…	…
126	126
127	127
128	128
…	…
253	255
254	255
255	255


Input Image


Output Image

Figure 93: Example 2 of enhancing contrast of an image using look-up table

Look-up table in CameraSuite SDK

In the *CameraSuite SDK* the *ImageProcAPI* provides the programming interface for generating and modifying look-up tables. The bit depth and image type supported are shown in Table 52. For a detailed description on how to use the look-up table feature please refer to the *CameraSuite API Help* located in the doc folder of the *CameraSuite SDK* installation directory.

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

Table 52: Look-up table - supported bit depth and image types

Look-up table in CameraSuiteClient

The *CameraSuiteClient* 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 *CameraSuite SDK* 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 94: User-defined XML file for 8 Bit RGB color (left) or monochrome (right) images

Figure 94 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 95 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*.

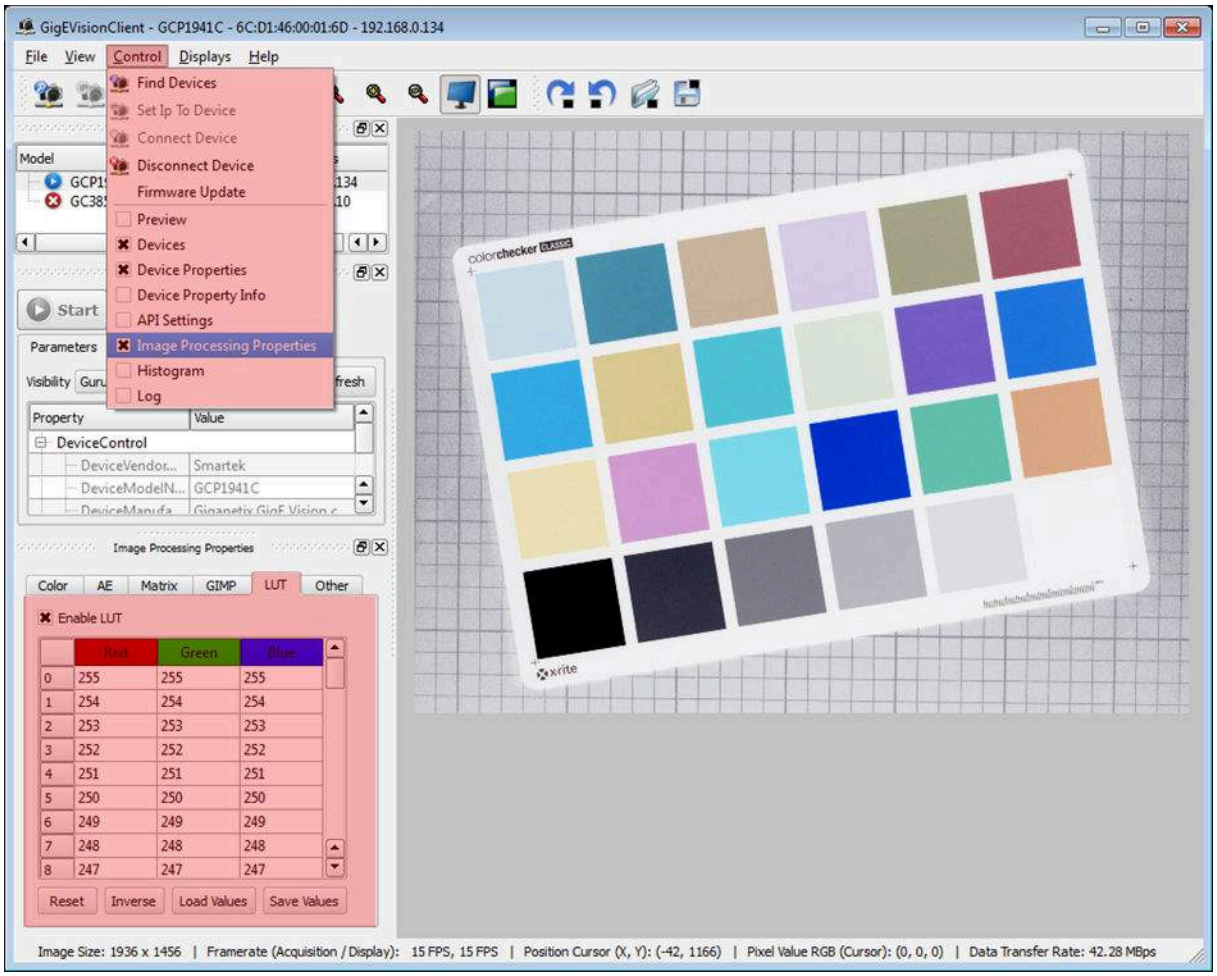


Figure 95: Look-up table feature in CameraSuiteClient

- **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

8.2.2 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:

$$\mathrm{Pixel(x,y)_{out}} = \mathrm{Pixel(x,y)_{In}} \times \mathrm{DigitalGain}$$

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 96 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 96: Digital Gain to brighten an image

Note

In contrast to the analog gain the digital gain produces "holes" in the histogram, shown in Figure 97. 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.

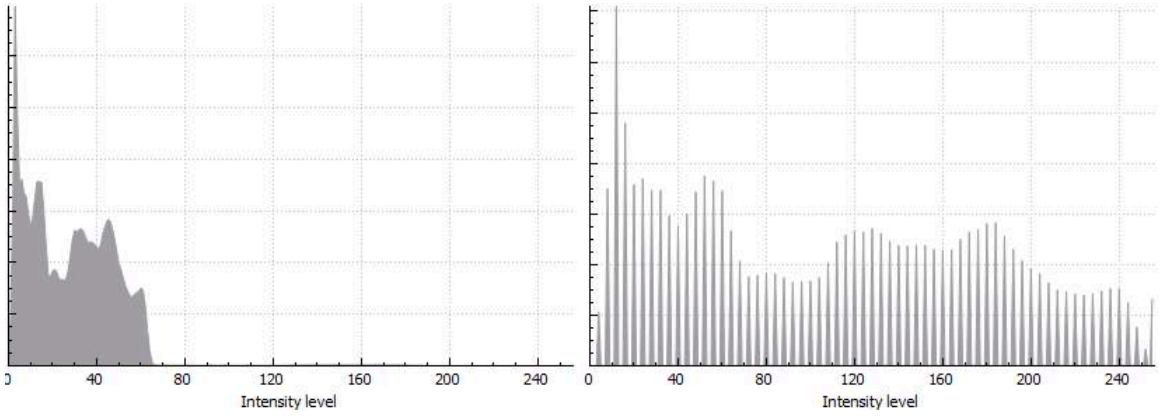


Figure 97: Digital Gain - Histogram original image (left) and after applying digital gain of 4.0 (right)

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 53. The analog gain is therefore always to be preferred where possible.

Pixel_{In}	$\text{Pixel}_{\text{Out}} = \text{Pixel}_{\text{In}} \times 2.0$
0	0
1	2
2	4
3	6

Table 53: Digital Gain - Output values

Digital Gain in CameraSuite SDK

In the *CameraSuite SDK* the *ImageProcAPI* provides the programming interface to apply digital gain to images. The bit depths and image types supported are shown in Table 54. For a detailed description on how to use the digital gain feature please refer to the *CameraSuite API Help* located in the doc folder of the *CameraSuite SDK* installation directory.

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

Table 54: Digital Gain - supported bit depth and image type

Digital Gain in the CameraSuiteClient

In the *CameraSuiteClient* the *Digital Gain* can be accessed in the *Image Processing Properties* panel under *Color / Mono*, shown in Figure 98. If not visible, the panel can be enabled by the menu bar entry *Control* ⇒ *Image Processing Properties*.

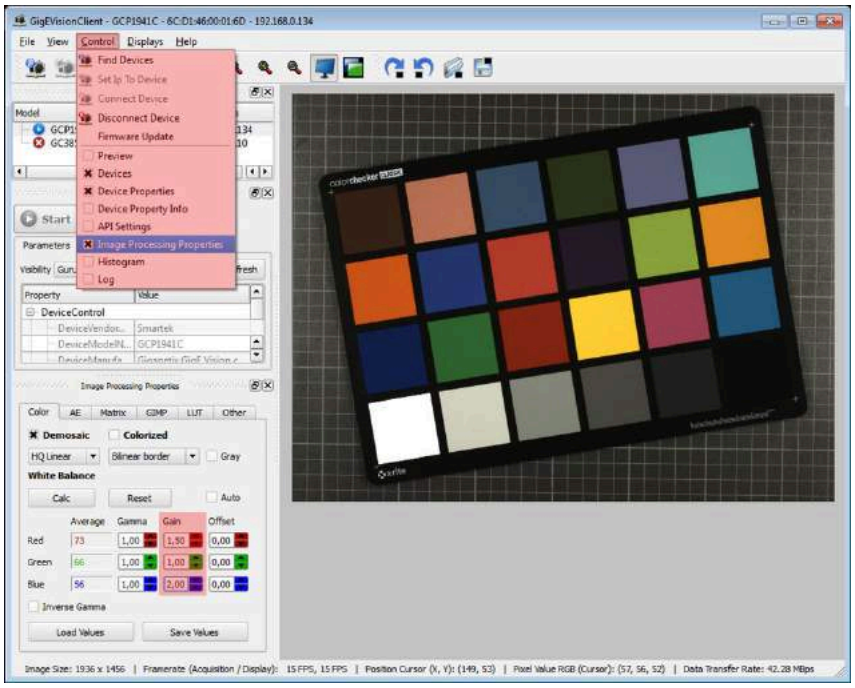


Figure 98: Digital Gain in CameraSuiteClient

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.

8.2.3 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 CameraSuite SDK

In the *CameraSuite SDK* 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 55. For a detailed description on how to use the auto exposure feature please refer to the *CameraSuite API Help* located in the doc folder of the *CameraSuite SDK* installation directory.

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

Table 55: Auto exposure - supported bit depth and image type

Auto Exposure in the CameraSuiteClient

In the *CameraSuiteClient* the *Auto Exposure* (AE) can be enabled / disabled in the *Image Processing Properties* panel under *AE* (see Figure 99). If not visible, the panel can be enabled by the menu bar entry *Control ⇒ 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 [µs] (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 [µs] (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.

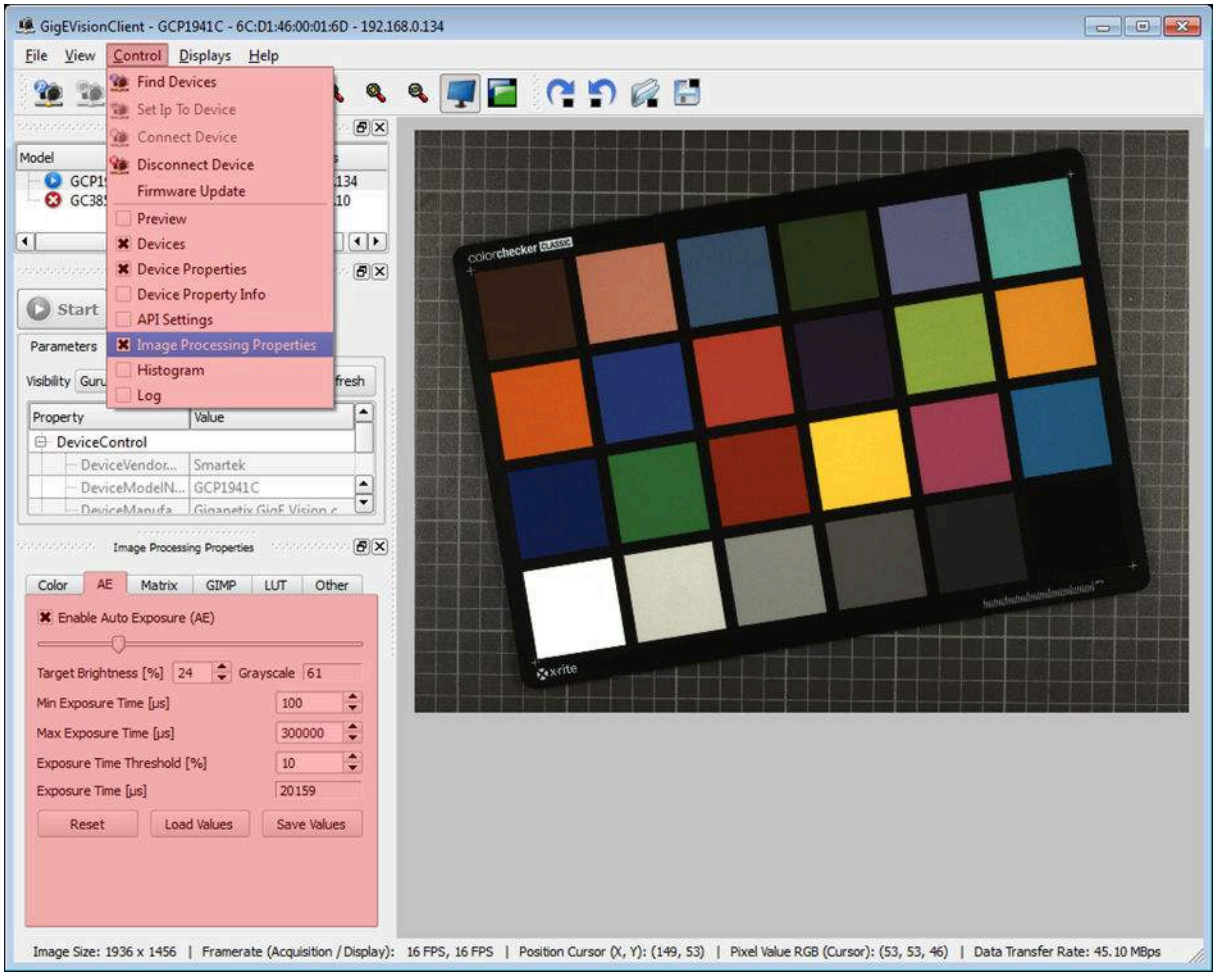


Figure 99: Auto Exposure in CameraSuiteClient.

8.2.4 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 100 illustrates the color temperature spectrum of visible light in the range from 1000K to 15000K.

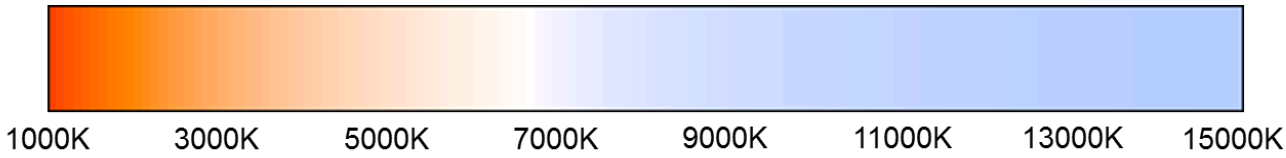


Figure 100: Color temperature spectrum in range 1000K to 15000K

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 101 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 101: 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 102 demonstrates the White Balancing feature implemented in the *CameraSuite SDK*. The green color cast is corrected, the output image appears as it should.

Figure 102: ColorChecker chart without (left) and with White Balancing (right)

White Balance in the CameraSuite SDK

In the *CameraSuite SDK* the *ImageProcAPI* provides the programming interface for executing the White Balance algorithm. The bit depth and image types supported are shown in Table 56

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

Table 56: 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 *CameraSuite API Help* located in the doc folder of the *CameraSuite SDK* installation directory.

White Balance in the CameraSuiteClient

In *CameraSuiteClient* the user can apply the *White Balance* algorithm once or repeatedly for every incoming frame. All options can be can be accessed in the *Image Processing Properties* panel under Color shown in Figure 103. 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 an applies correction gains for every frame. This mode is recommended when the lighting condition may permanently change.

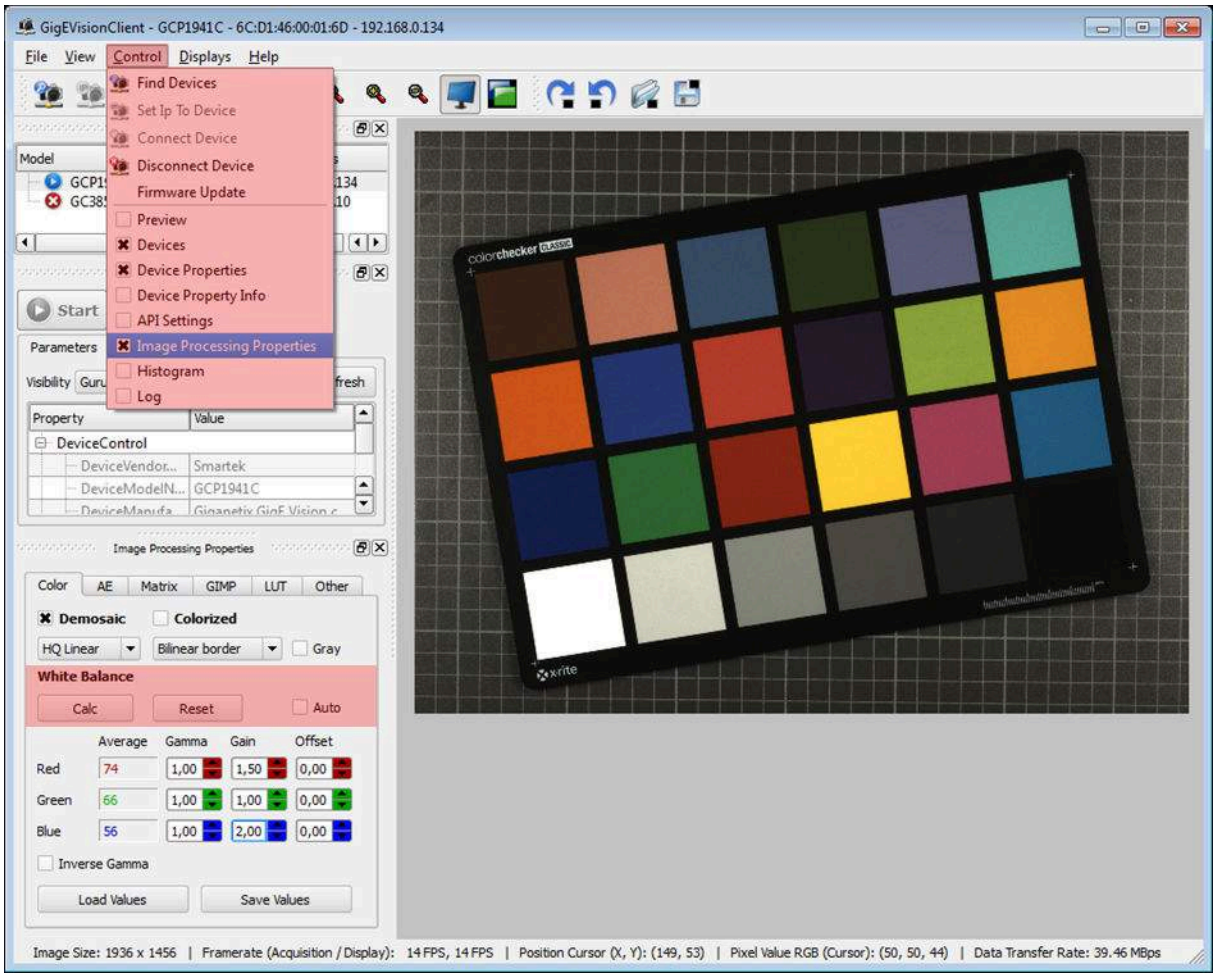


Figure 103: White Balance feature in the CameraSuiteClient

- **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.

8.2.5 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 104. 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 104 (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.

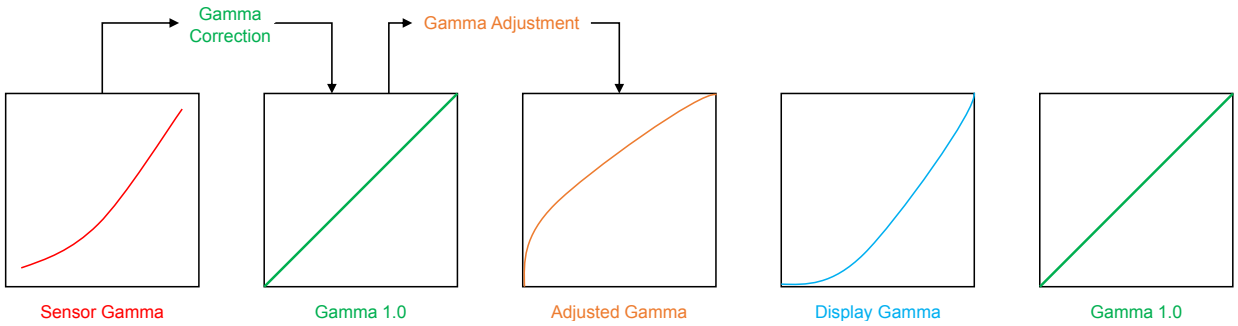


Figure 104: Gamma correction workflow

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:

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

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 CameraSuite SDK

In the *CameraSuite SDK* 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 57. For a detailed description on how to use the digital offset, gain and gamma correction feature please refer to the *CameraSuite API Help* located in the doc folder of the *CameraSuite SDK* installation directory.

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

Table 57: Gamma Correction - supported bit depth and image type

Gamma Correction in CameraSuiteClient

In the *CameraSuiteClient* Gamma, Gain and Offset can be accessed in the *Image Processing Properties* panel under *Color / Mono*, shown in Figure 105. If not visible, the panel can be enabled by the menu bar entry *Control ⇒ Image Processing Properties*.

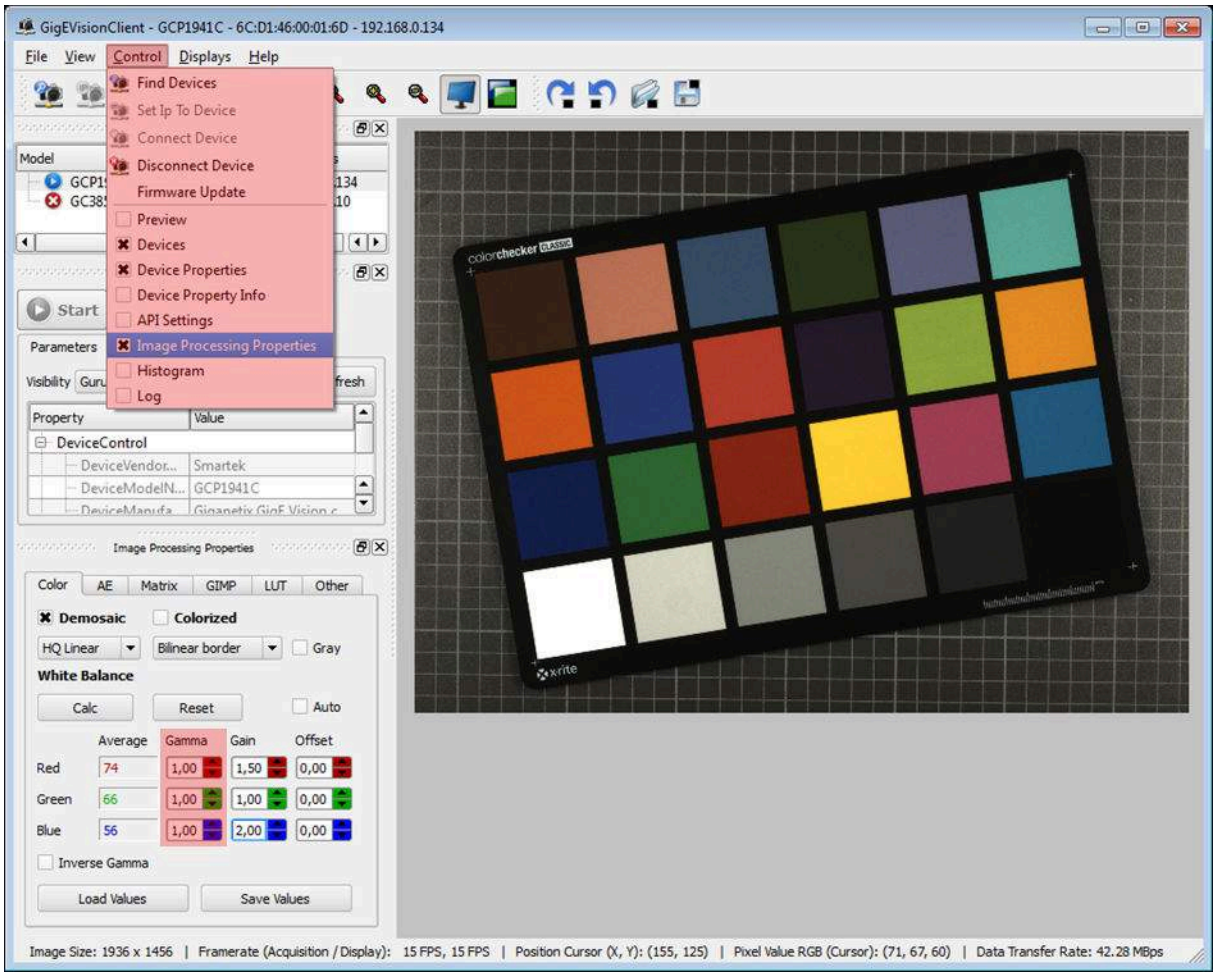


Figure 105: Gamma Correction dialog

8.2.6 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 2×2 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 106 shows the "GR" filter alignment, which means that the pattern starts with green (G) followed by red (R).

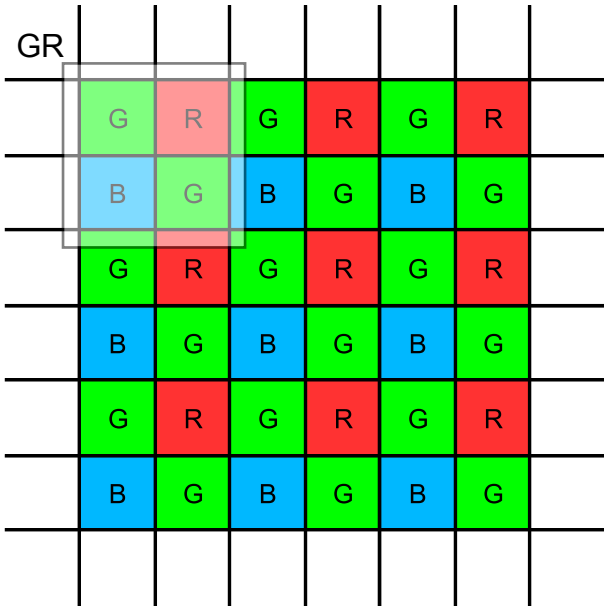


Figure 106: Bayer Filter Pattern GR

The illustration in Figure 106 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 107.

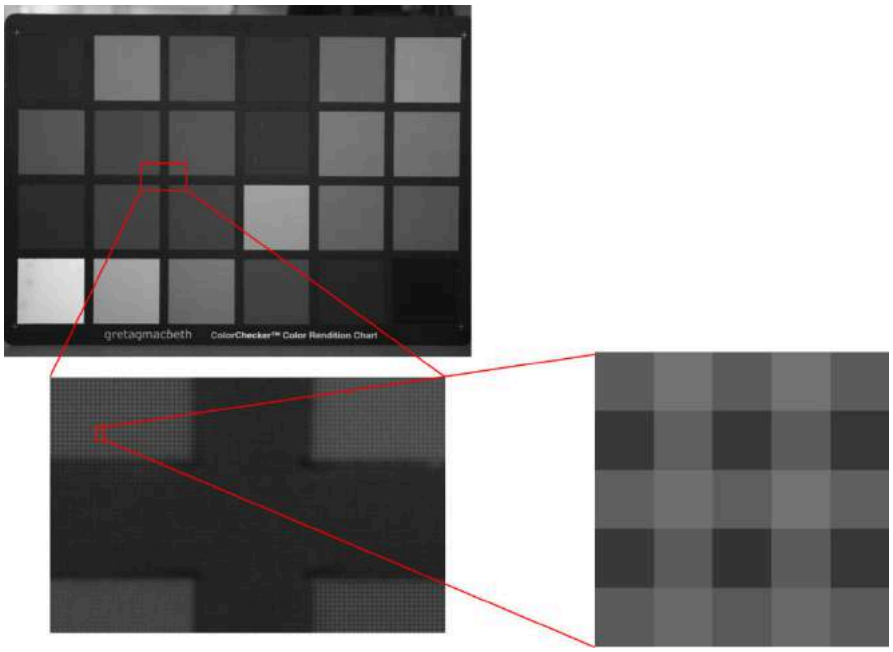


Figure 107: Raw image overlaid with a Bayer pattern

Figure 107 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 *CameraSuite SDK* 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 108 and Figure 109 for *Bilinear* and *HQ Linear*.

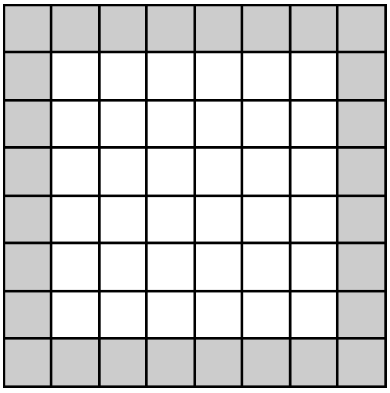


Figure 108: Bilinear algorithm border

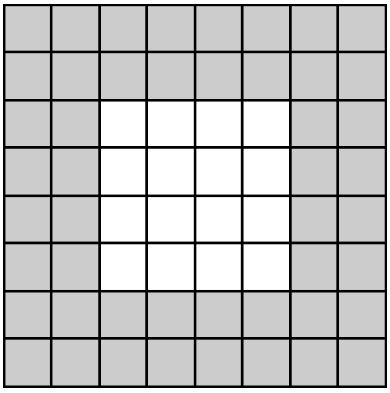


Figure 109: HQ Linear algorithm border

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 CameraSuite SDK

In the *CameraSuite SDK* 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 58. For a detailed description on how to use the demosaicing feature please refer to the *CameraSuite API Help* located in the doc folder of the *CameraSuite SDK* installation directory.

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

Table 58: Demosaicing - supported bit depth and image type

Demosaicing in CameraSuiteClient

In the *CameraSuiteClient* the demosaicing options can be accessed in the *Image Processing Properties panel* under *Color*, shown in Figure 110. If not visible, the panel can be enabled by the menu bar entry *Control ⇒ Image Processing Properties*.

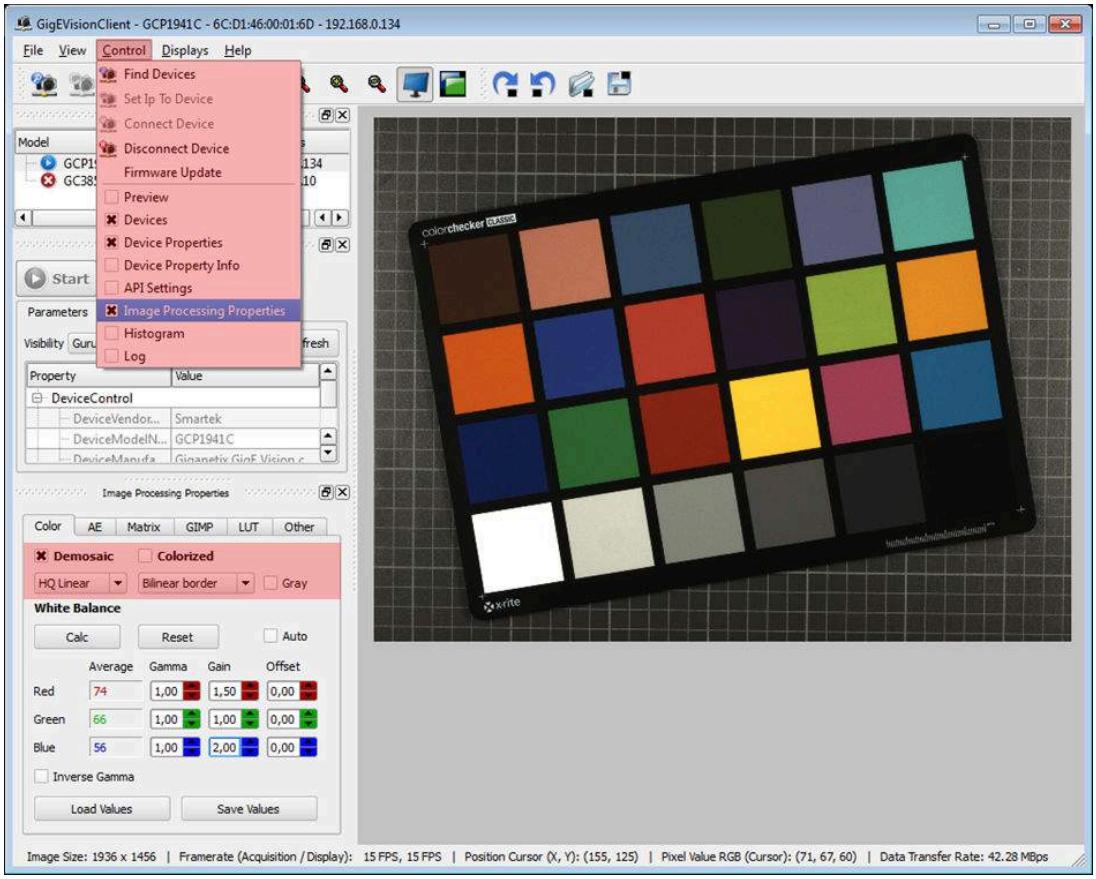


Figure 110: Demosaicing algorithms with border type selection

8.2.7 Matrix Multiplication 3x3

N-by-N matrices are commonly used to transform RGB colors, scale them and control hue, saturation and contrast. The *CameraSuite SDK* 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 111 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 111: 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 CameraSuite SDK

In the *CameraSuite SDK* 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 59.

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

Table 59: Matrix Multiplication - supported bit depth and image type

Matrix Multiplication 3x3 in the CameraSuiteClient

In the *CameraSuiteClient* the demosaicing options can be accessed in the *Image Processing Properties* panel under Matrix, shown in Figure 112. If not visible, the panel can be enabled by the menu bar entry *Control* ⇒ *Image Processing Properties*.

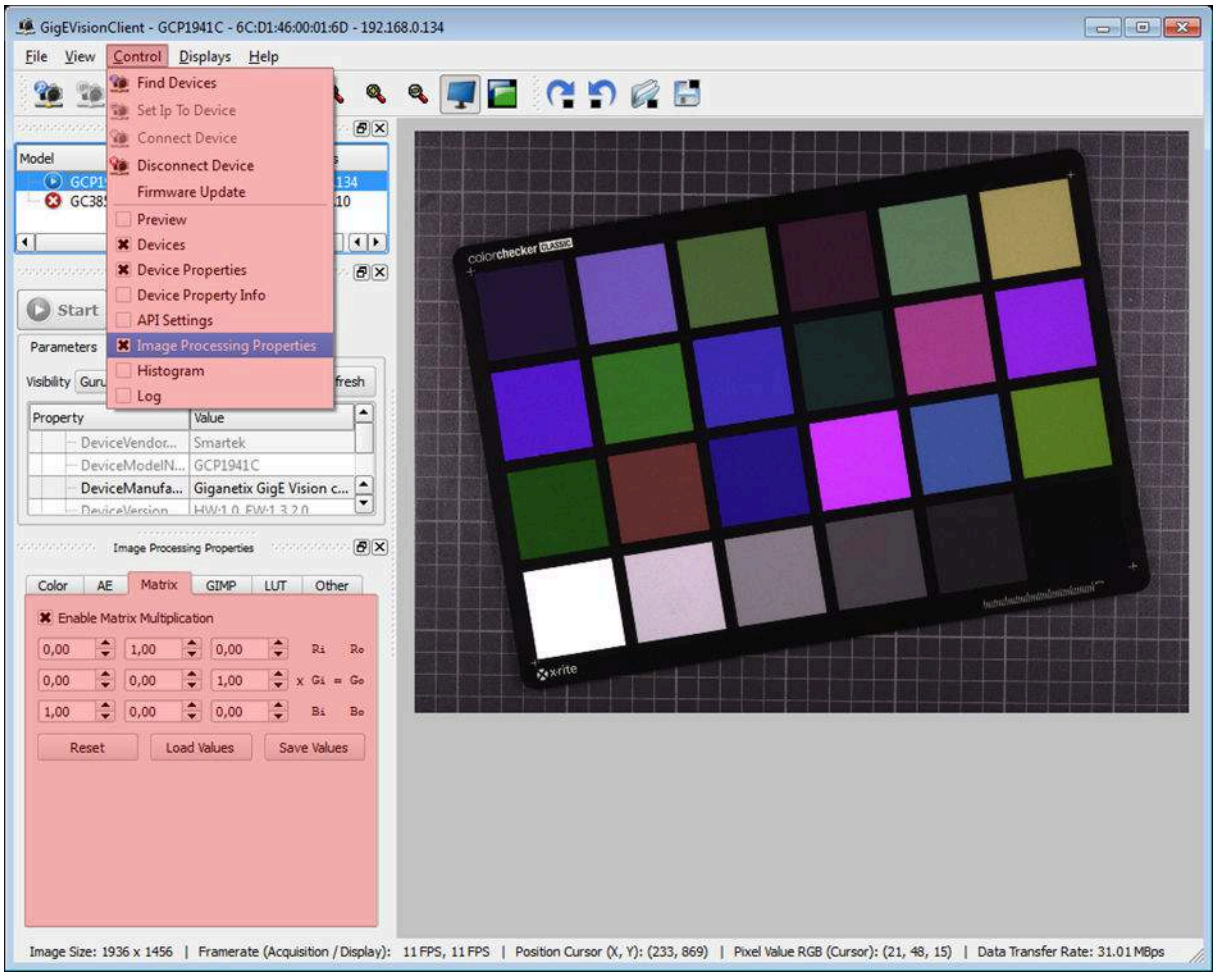


Figure 112: Matrix Multiplication RGB in the CameraSuiteClient

- **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

8.2.8 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 113.

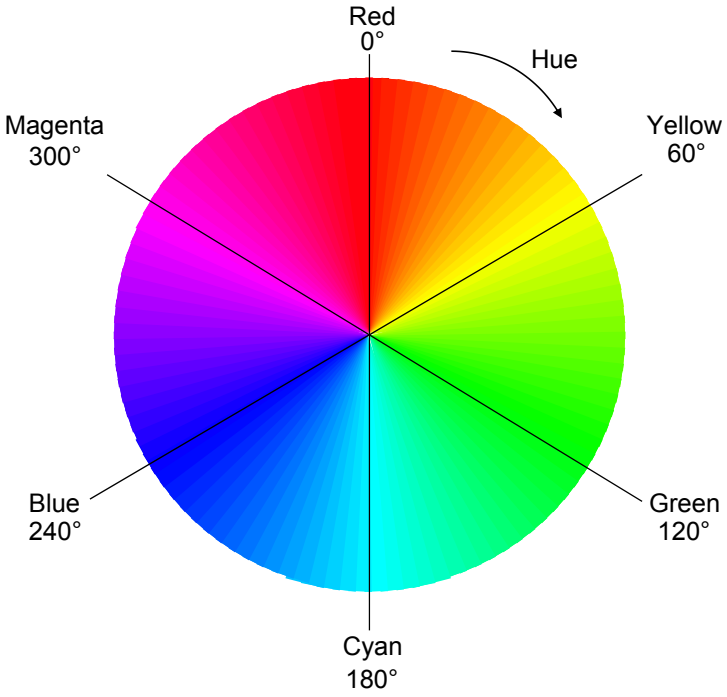


Figure 113: HSL color wheel

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 60 shows the six base colors.

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

Table 60: HSL color space

As shown in Figure 113 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 CameraSuite SDK

In the *CameraSuite SDK* the *ImageProcAPI* provides the programming interface for configuring and executing *Gimp HSL* algorithm. The bit depth and image type supported are shown in Table 61. For a detailed description on how to use this feature please refer to the *CameraSuite API Help* located in the doc folder of the *CameraSuite SDK* installation directory.

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

Table 61: Gimp HSL - supported bit depth and image type

Gimp HSL in the CameraSuiteClient

In the *CameraSuiteClient* the *GIMP HSL* manipulation options can be accessed in the *Image Processing Properties* panel under GIMP, shown in Figure 114. If not visible, the panel can be enabled by the menu bar entry *Control* ⇒ *Image Processing Properties*. If Master is selected, then values are changed for every channel at once.

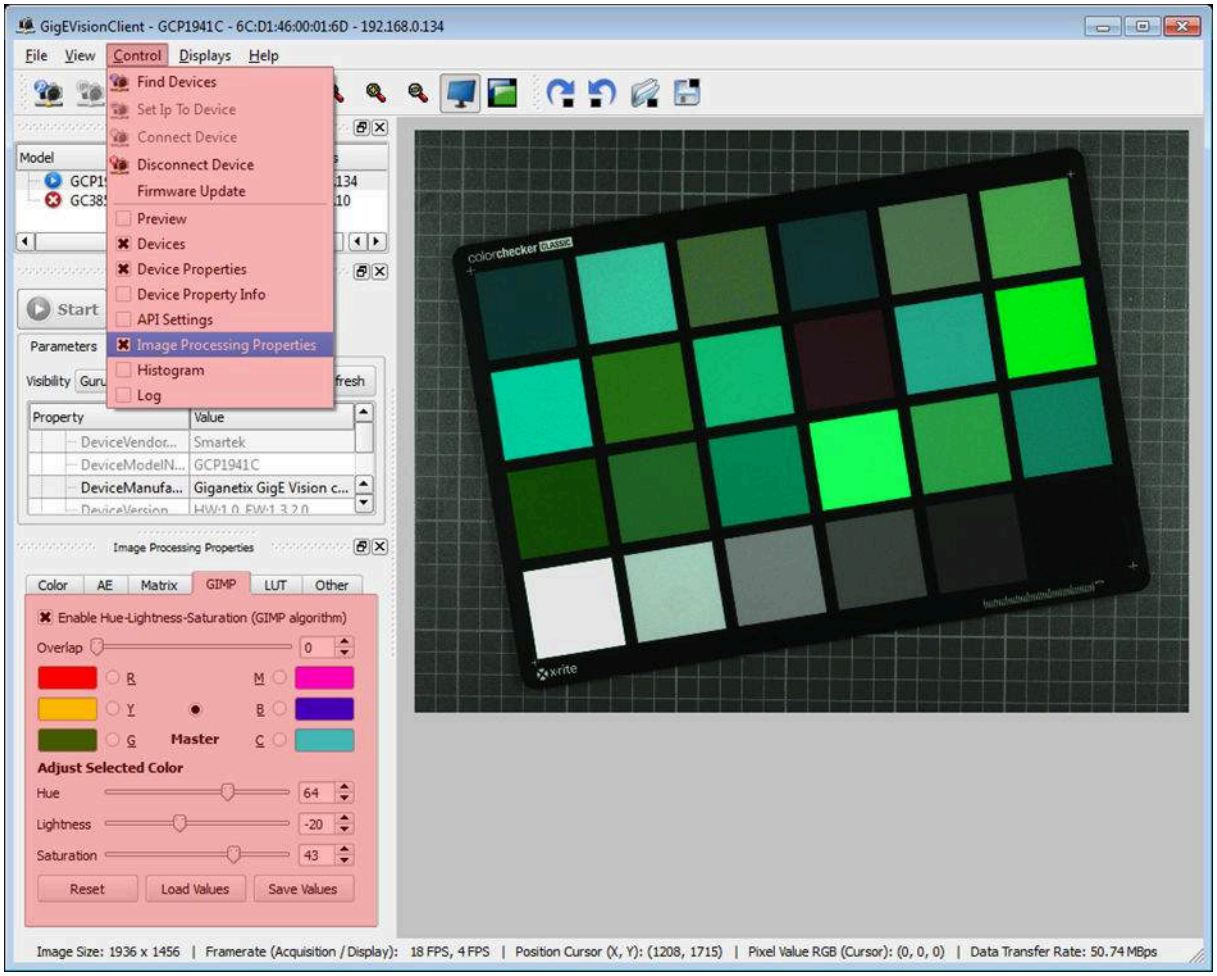


Figure 114: Color GIMP dialog

- **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

8.2.9 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 115 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.

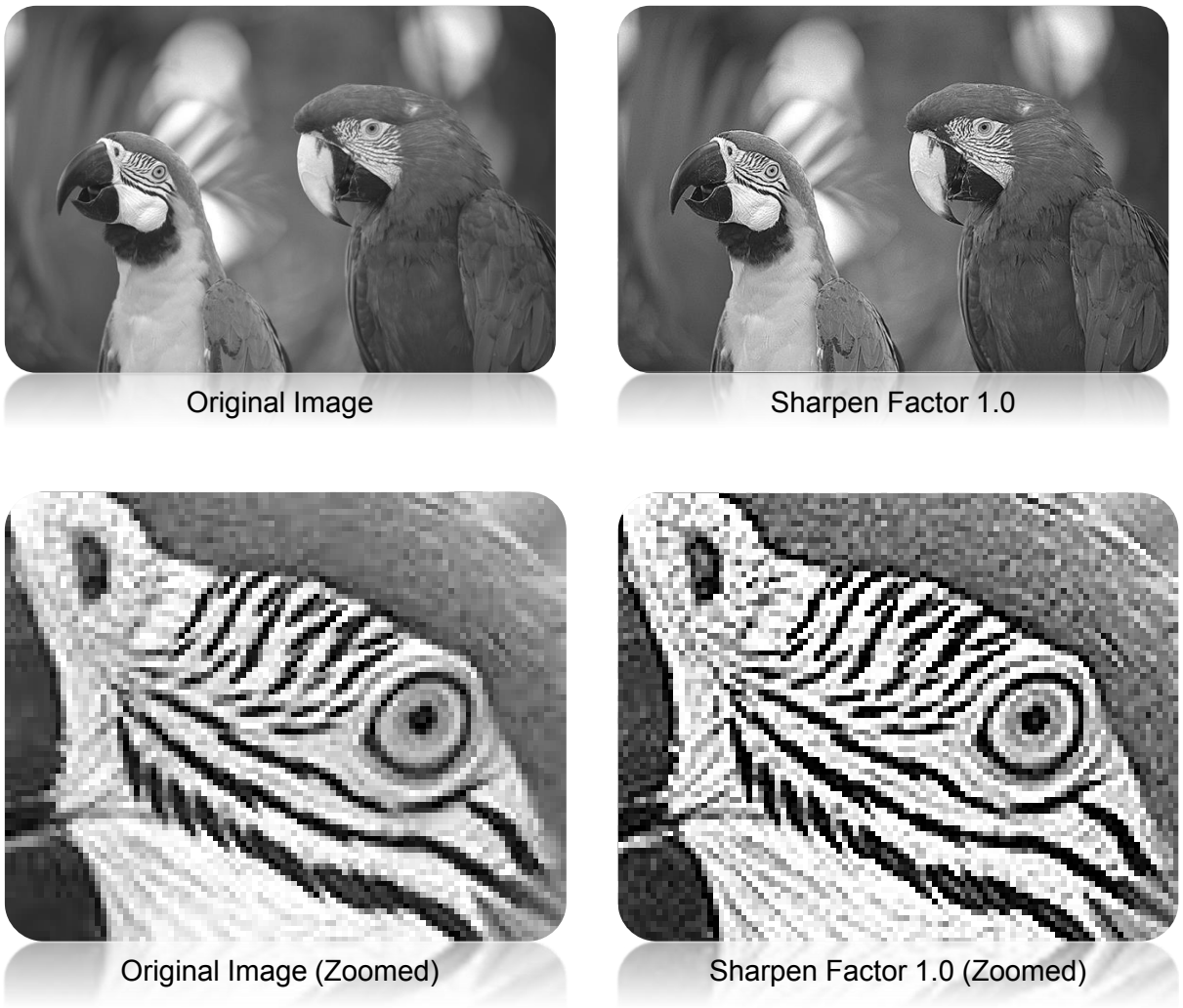


Figure 115: Example of sharpening algorithm (factor 1.0)

Sharpening in the CameraSuite SDK

In the *CameraSuite SDK* the *ImageProcAPI* provides the programming interface for configuring and executing the sharpening algorithm. The bit depths and image types supported are shown in Table 62. For a detailed description on how to use this feature please refer to the *CameraSuite API Help* located in the doc folder of the *CameraSuite SDK* installation directory.

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

Table 62: Sharpening - supported bit depth and image type

Sharpening in the CameraSuiteClient

In the *CameraSuiteClient* the *Image Sharpen* options can be accessed in the *Image Processing Properties* panel under *Other*, shown in Figure 116. If not visible, the panel can be enabled by the menu bar entry *Control* ⇒ *Image Processing Properties*.

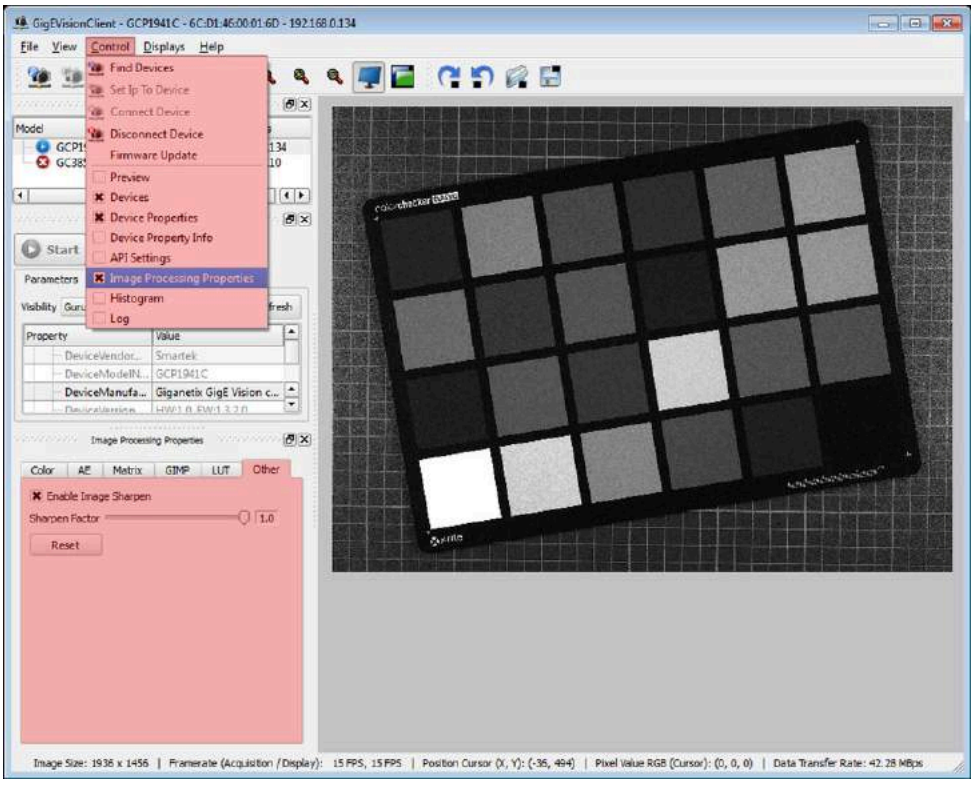


Figure 116: Sharpening in the CameraSuiteClient

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

8.2.10 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 117 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.

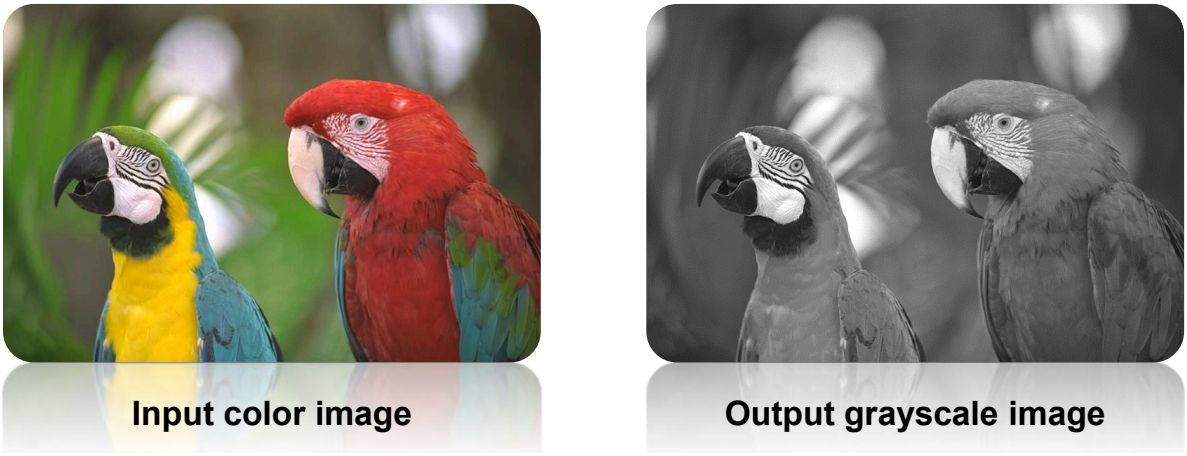


Figure 117: Example of RGB-to-gray conversion

RGB-to-Gray Conversion in the CameraSuite SDK

In the *CameraSuite SDK* the *ImageProcAPI* provides the programming interface for executing the RGB-to-gray conversion. The bit depths and image types supported are shown in Table 63. For a detailed description on how to use this feature please refer to the *CameraSuite API Help* located in the doc folder of the *CameraSuite SDK* installation directory.

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

Table 63: RGB to Gray conversion - supported bit depths and image types

RGB-to-Gray Conversion in the CameraSuiteClient

In the *CameraSuiteClient* the RGB-to-Gray options can be activated in the *Image Processing Properties* panel under *Color*, shown in Figure 118. If not visible, the panel can be enabled by the menu bar entry *Control* ⇒ *Image Processing Properties*.

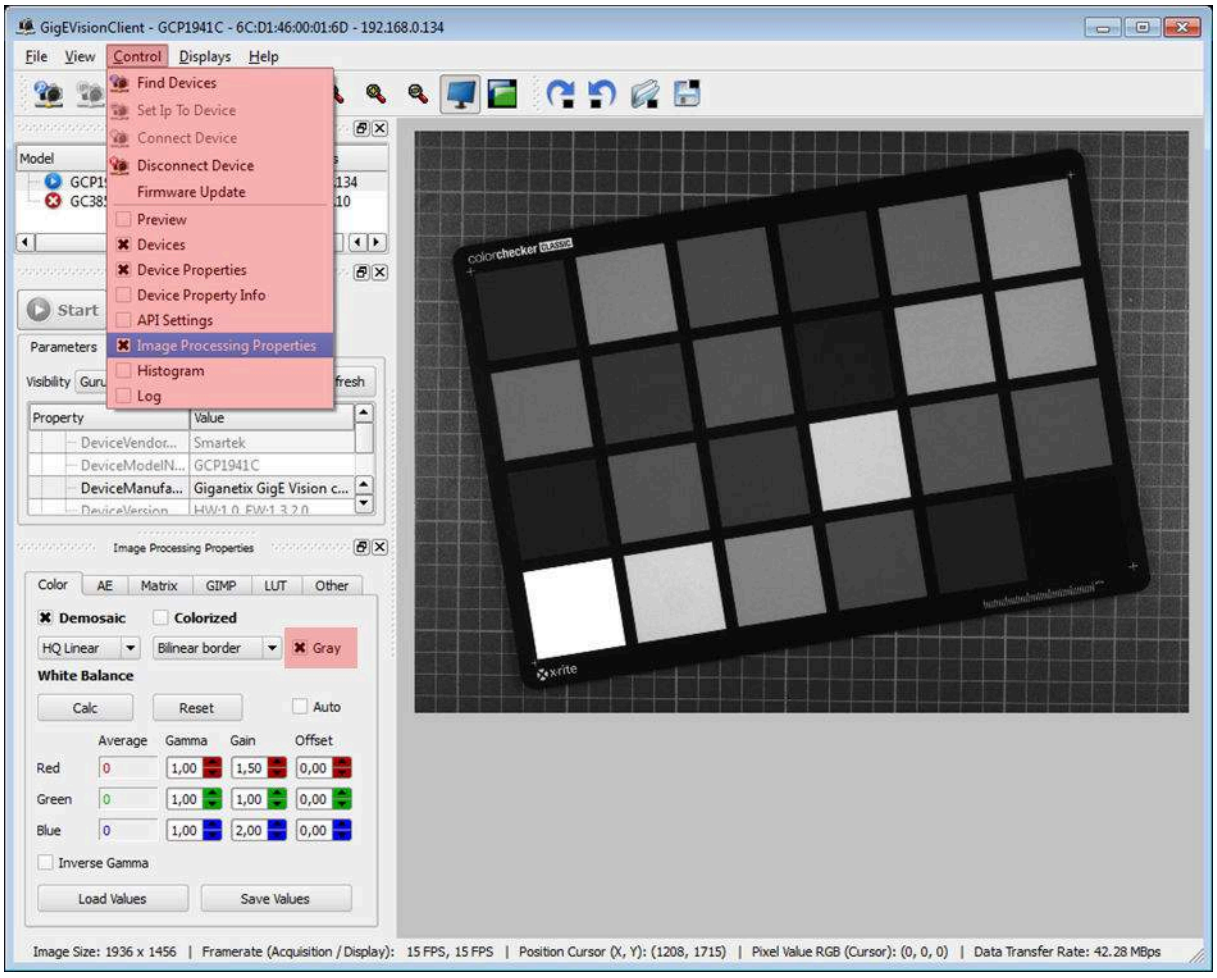


Figure 118: RGB to Gray conversion in CameraSuiteClient

8.2.11 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 CameraSuite SDK

In the *CameraSuite SDK* 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 64.

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

Table 64: Bit depth conversion - supported bit depth and image type

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

Bit Depth Conversion in the CameraSuiteClient

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 *CameraSuiteClient*.

8.2.12 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 θ 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 119 demonstrates all flipping cases and Figure 120 demonstrates all rotation cases.

Figure 119: Example of image flipping

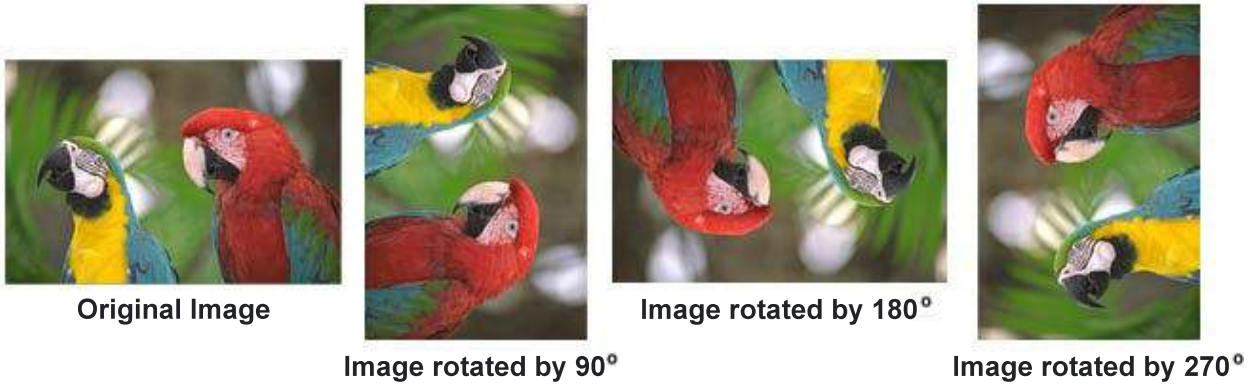


Figure 120: Example of image rotation

Flipping / Rotating in the CameraSuite SDK

In the *CameraSuite SDK* the *ImageProcAPI* provides the programming interface for executing the flipping / rotating transformations. The bit depths and image types supported are shown in Table 65. For a detailed description on how to use this feature please refer to the *CameraSuite API Help* located in the doc folder of the *CameraSuite SDK* installation directory.

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

Table 65: Flip - Rotate / supported bit depths and image types

Flip / Rotate transformations in the CameraSuiteClient

In the *CameraSuiteClient* the Flip / Rotate options can be activated in the *Image Processing Properties* panel under *Color*, shown in Figure 121. If not visible, the panel can be enabled by the menu bar entry *Control* ⇒ *Image Processing Properties*.

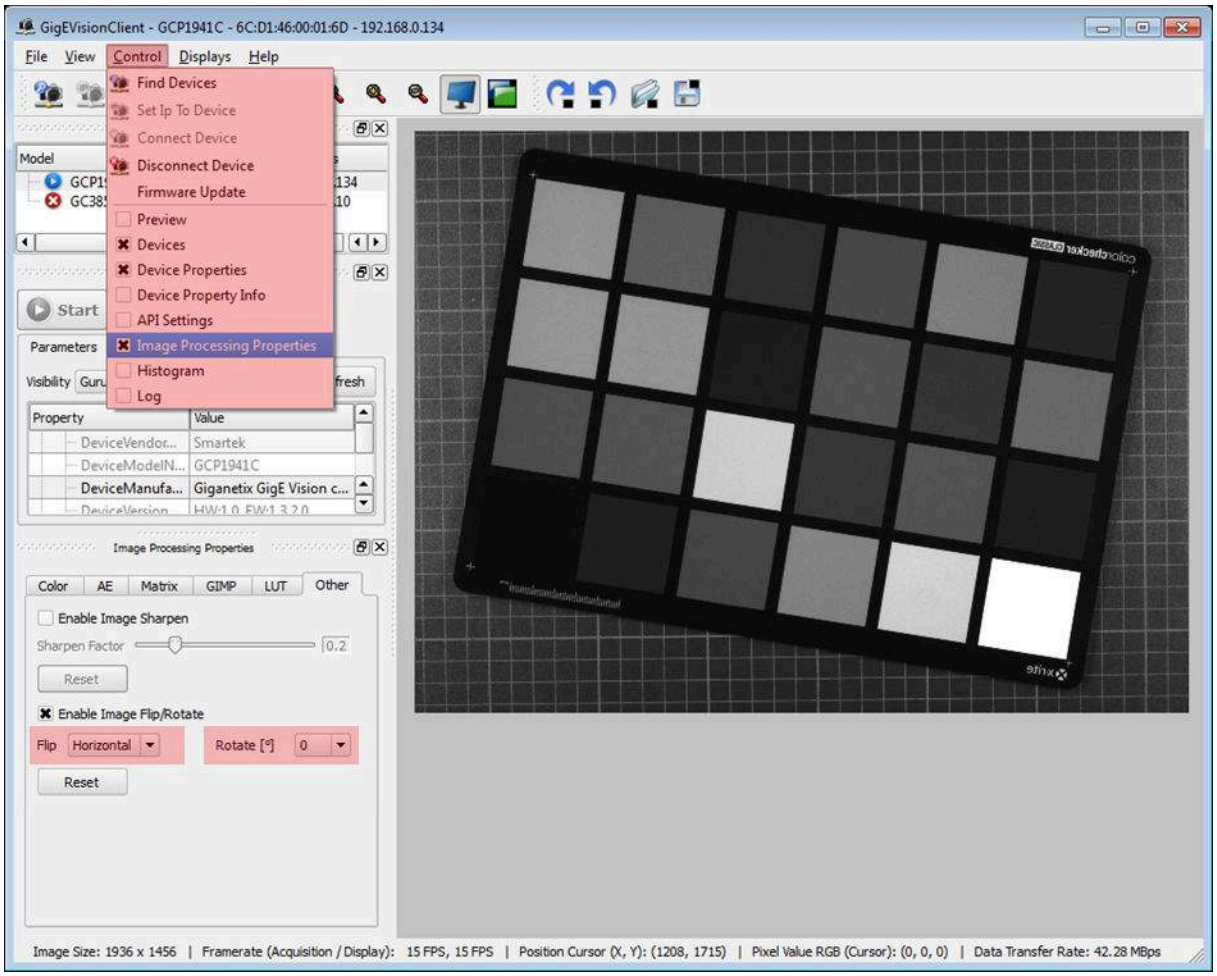


Figure 121: Flip / Rotate transformations in CameraSuiteClient

8.3 Color Image Processing Pipeline

In the previous chapters the image processing algorithms provided by the *ImageProcAPI* in of the *CameraSuite SDK* 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 122.

Figure 122: Image Processing Pipeline

The order of the algorithms in the color image pipeline provided by the *CameraSuite SDK* 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 by combined by the available algorithms in a preferred order.

Color Image Processing Pipeline in CameraSuite SDK

In the *CameraSuite SDK* 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 66. For a detailed description on how to use this feature please refer to the *CameraSuite API Help* located in the doc folder of the *CameraSuite SDK* installation directory.

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

Table 66: Color pipeline - supported bit depth and supported image type

Color Image Processing Pipeline in CameraSuiteClient

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.

9 Revision History

Ver.	Chapter	Changes	Date
1.0.2	All	Initial Release (Final)	2017-07-03
1.0.1	All	Initial Release (Preliminary)	2017-05-19

Table 67: Revision History

10 Contact Information

Published by:

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

www.SMARTEK.vision

Email: info@SMARTEKvision.com
Tel: +385 (40) 493 805
Fax: +385 (40) 493 819