ximea

xiSpec2

- hyperspectral imaging camera series

Technical Documentation

Version 2.00, July 2021

1. Introduction

1.1. About This Manual

Dear customer,

Thank you for purchasing a hyperspectral imaging product from XIMEA.

We hope that this technical manual, together with the manuals for the xiQ- and xiX-camera series, can answer your questions, but should you have any further questions or if you wish to claim a service or warranty case, please contact your local dealer or refer to the XIMEA Support on our website: www.ximea.com/support

This document is subject to change without notice.

1.1.1. Contact XIMEA

XIMEA is a worldwide operating company.

Headquarters
Sales worldwide

XIMEA GmbH
Am Mittelhafen 16
48155 Münster
Germany

Tel: +49 (251) 202 408-0
Fax: +49 (251) 202 408-99

Sales Americas

XIMEA Corp.
12600 W Colfax Ave., Suite A-130
Lakewood, CO 80215
USA

Tel: +1 (303) 389-9838
Fax: +1 (303) 202-6350

R&D, Production

XIMEA S.R.O.
Lesna 52
900 33 Marianka
Slovakia

Internet	www.ximea.com
General inquiries	info@ximea.com
Sales	sales@ximea.com
Support	support@ximea.com

1.2. Helpful Links

XIMEA Homepage	https://www.ximea.com
xiSpec hyperspectral cameras	https://www.ximea.com/en/usb3-vision-camera/hyperspectral-usb3-cameras-mini
USB3 Camera Zone	https://www.ximea.com/en/usb3-vision-camera/usb3zone
XIMEA API / SDK description	https://www.ximea.com/support/wiki/apis/APIs
xiAPI stable versions download	https://www.ximea.com/support/documents/4
xiAPI beta versions download	https://www.ximea.com/support/documents/14
Frequently Asked Questions	https://www.ximea.com/support/wiki/standard-cameras/FAQ_Hyperspectral_cameras
Knowledge Base	https://www.ximea.com/support/wiki/allprod/Knowledge_Base
XIMEA Registration	https://www.ximea.com/en/products/register
XIMEA Terms & Conditions	https://www.ximea.com/en/corporate/generaltc

1.2.1. Table of Contents

2. xiSpec2 Camera Series

2.1. What is xiSpec2?

xiSpec2 is an ultra-compact Industrial hyperspectral imaging (HSI) camera family.

The housing has been optimized for improved spectral results. This optimization is achieved by a new special coating on the inside of the camera. The coating is very sensitive and should not be touched.

Special bandpass filters were newly developed. The filters are now built into the camera.

The whole camera setup (camera, camera specific bandpass filters, standard lenses) is calibrated together – not only the sensor on wafer level. The calibration results (calibration file and calibration test report) are delivered with the cameras. The lens family which is used for system calibration is part of the starter kit.

It is recommended to use the lenses which are part of the starter kits. These lenses are used for system calibration.

The HSI sensors used are manufactured by imec, a Belgian research institution. Imec and XIMEA cooperate closely in development, sales and customer support for the xiSpec2 hyperspectral cameras. Both the cameras and the starter kits are co-branded.

2.2. Facts and Features

The xiSpec2 camera series has outstanding facts and features:

Facts

- Cameras with 16 to 150 spectral bands
- 170 fps with USB3 interface, 340 fps with PCIe
- Snapshot and line scan versions
- Smallest and lightest hyperspectral cameras available, dimensions of 26.4 x 26.4 x 30.2 mm, mass of 32 grams (USB3)
- Low power consumption, typical only 1.5W (USB3)
- Rugged, without moving parts
- Each camera is spectrally calibrated

Features

- Small - fast - flexible
- Snapshot with global shutter
    - 4x4 with 16 bands in the visible 460-600 nm range
    - 4x4 with 15 bands in the red / NIR 600-860 nm range
    - 5x5 with 24 bands in the NIR 665-960 nm range
- Multi-line scan with global shutter
    - 150+ spectral bands 470-900 nm range
- Starter kits available for rapid development
- Flexible and programmable GPIO options

2.3. Models Overview, sensors and models

The sensor technology used in XIMEA's xiSpec2 HSI-cameras (HSI = Hyperspectral imaging) is based on standard CMOS area sensors with a native resolution of 2048*1088 pixels (AMS/CMOSIS CMV2000 mono).

The HSI sensors are integrated either in the standard USB3 camera series xiQ with up to 170 frames per second or in the standard PCIe 2 lane Gen 2 camera series xiX with up to 340 frames per second.

The technical manual for the xiQ camera series is available at:

https://www.ximea.com/downloads/usb3/manuals/xiq_technical_manual.pdf

The technical manual for the xiX camera series is available at:

https://www.ximea.com/downloads/cb/manuals/xix_technical_manual.pdf

The xiSpec2 cameras are based on the camera model MQ022MG-CM (xiQ) or MX022MG-CM-X2G2-Fx (xiX), but of course use other sensors and special filter glasses to optimally support the HSI sensors and calibration data specially adapted for the xiSpec2 cameras. All standard camera parameters (structure, dimensions, interface, IO system, ...) can be found in the xiQ or xiX manual.

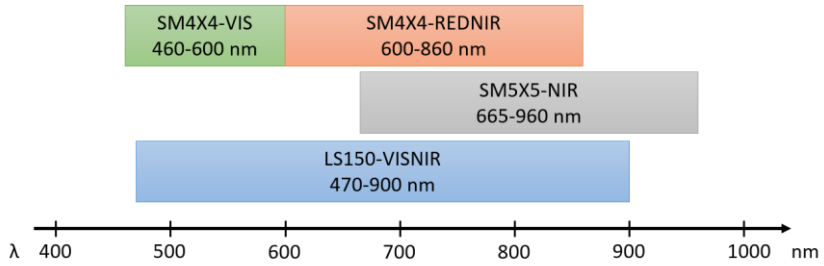
Basic information about the camera

Value	Description
Sensor	AMS / CMOSIS CMV2000 mono
Resolution	2048 x 1088, 2.2 Mpix
Pix. size [µm]	5.5
ADC [bits]	10, 8
Sensor size / diagonal [mm]	11.3 x 6.0
Optical size	2/3 "
Fps	170 (USB3.0), 340 (PCIe X2G2)
Power typ. [W]	1.5 (xiQ)

table 2-1, basic camera specs

Note: Maximum frame rate measured at 8 bits per pixel

An overview of the xiSpec2 camera models is shown in the following diagram:



2.4. Model Nomenclature

Order numbers name conventions for the different models:

<Series>022HG-IM-<Layout>-<Wavelength>[-<Options>]

<Series>: camera series

MQ: xiQ USB3 camera family

MX: xiX PCIe camera family

<Layout>: Layout of the Interference filter

SMnXn snapshot mosaic layout, n=4, 5

LSm line scan layout, m= 150

<Options>: Connector options

(none) : micro-B USB3 (standard)

FL: USB3 or PCIe flex-line connection horizontal

FV: PCIe flex-line connection vertical

FF: PCIe FireFly connection

Model	Spectral Bands	Spectral range [nm]	Interface
MQ022HG-IM-LS150-VN2	150	470 – 900	USB3 micro-B (Standard)
MQ022HG-IM-SM4X4-RN2	15	600 – 860	USB3 micro-B (Standard)
MQ022HG-IM-SM4X4-VIS3	16	460 – 600	USB3 micro-B (Standard)
MQ022HG-IM-SM5X5-NIR2	24	665 – 960	USB3 micro-B (Standard)
MQ022HG-IM-LS150-VN2-FL	150	470 – 900	USB3 flex-cable
MQ022HG-IM-SM4X4-RN2-FL	15	600 – 860	USB3 flex-cable
MQ022HG-IM-SM4X4-VIS3-FL	16	460 – 600	USB3 flex-cable
MQ022HG-IM-SM5X5-NIR2-FL	24	665 – 960	USB3 flex-cable
MX022HG-IM-LS150-VN2-FL	150	470 – 900	PCIe 2lane Gen 2 flex-cable horizontal
MX022HG-IM-SM4X4-RN2-FL	15	600 – 860	PCIe 2lane Gen 2 flex-cable horizontal
MX022HG-IM-SM4X4-VIS3-FL	16	460 – 600	PCIe 2lane Gen 2 flex-cable horizontal
MX022HG-IM-SM5X5-NIR2-FL	24	665 – 960	PCIe 2lane Gen 2 flex-cable horizontal
MX022HG-IM-LS150-VN2-FV	150	470 – 900	PCIe 2lane Gen 2 flex-cable vertical
MX022HG-IM-SM4X4-RN2-FV	15	600 – 860	PCIe 2lane Gen 2 flex-cable vertical
MX022HG-IM-SM4X4-VIS3-FV	16	460 – 600	PCIe 2lane Gen 2 flex-cable vertical
MX022HG-IM-SM5X5-NIR2-FV	24	665 – 960	PCIe 2lane Gen 2 flex-cable vertical
MX022HG-IM-LS150-VN2-FF	150	470 – 900	PCIe 2lane Gen 2 FireFly
MX022HG-IM-SM4X4-RN2-FF	15	600 – 860	PCIe 2lane Gen 2 FireFly
MX022HG-IM-SM4X4-VIS3-FF	16	460 – 600	PCIe 2lane Gen 2 FireFly
MX022HG-IM-SM5X5-NIR2-FF	24	665 – 960	PCIe 2lane Gen 2 FireFly

table 2-2, models overview

3. General Overview

3.1. Spectral filters and filter response

Hyper spectral filters are added at wafer-level on top of the pixel structure of the sensor.

The spectral filters are Fabry-Perot interference filters. The thickness L of a cavity between two highly reflective surfaces and the refractive index n of the material in the cavity defines the main central wavelength λ of the filter.

$$k\lambda = 2nL\cos\theta$$

The response depends on

- The angle θ inside the cavity / between the light and perpendicular to the sensor surface
- The harmonics k of the interference

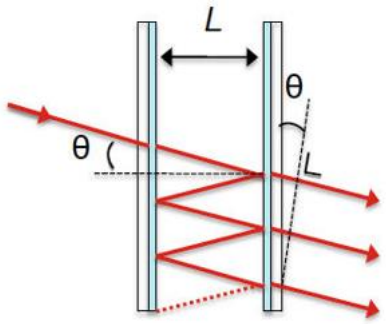


Figure 3-1, Fabry-Perot interference filter – schematics (©imec)

These filters are added to any of the pixels of the sensor individually (2048 * 1088 pixels with a size of 5.5 µm each).

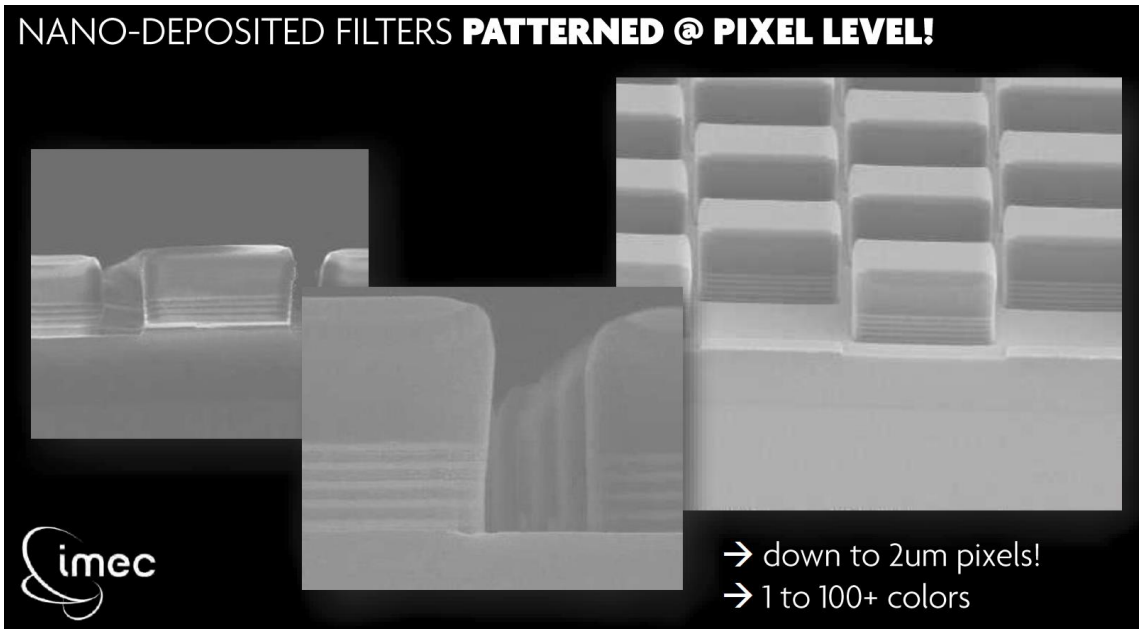


Figure 3-2, Microscope image at wafer level (©imec)

The response of an ideal interference filter has narrow peaks around the harmonic central wavelength. The intensity distribution is a Gaussian distribution with a given FWHM (Full Width at Half Maximum).

Different cavity heights used on a sensor result in different central wavelength which can be differentiated.

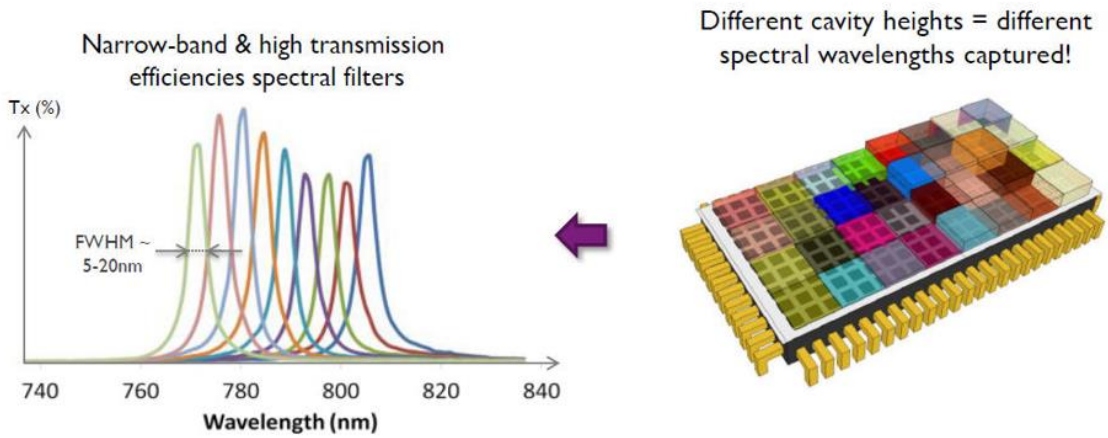


Figure 3-3, example: position of the Fabry-Perot filters and their ideal filter response

The following figure shows an example of the response curve of a 5x5 snapshot mosaic sensor.

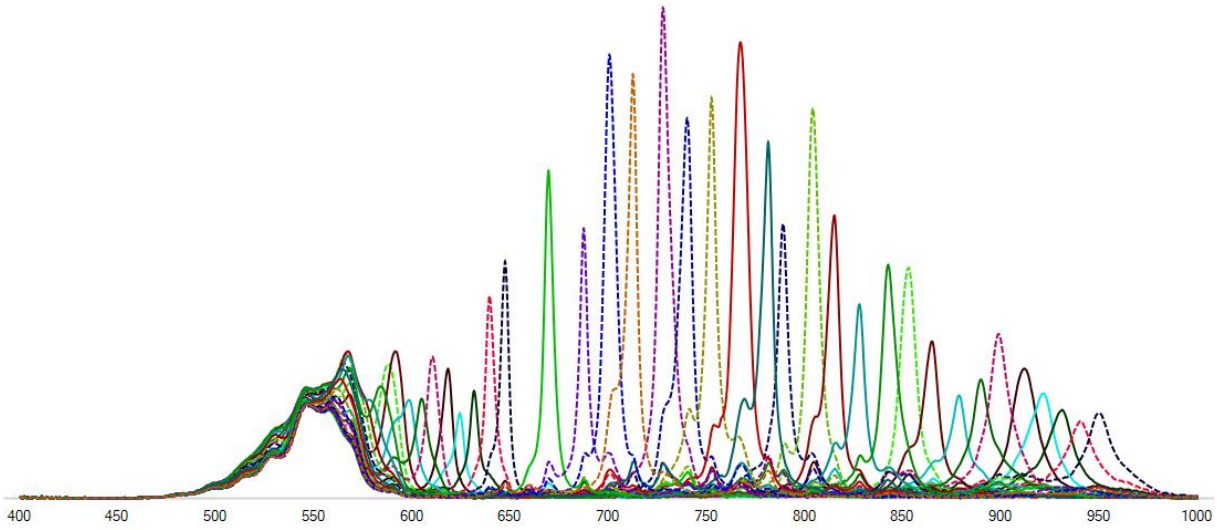


Figure 3-4, example: response / quantum efficiency curve of a snapshot mosaic sensor SM5X5

Some filters have two sensitivity peaks (first and second order response). Crosstalks happen when the signal of one pixel influences the signal on another (neighboring) pixel. This effect causes unwanted response outside the expected peaks of the influenced pixels.

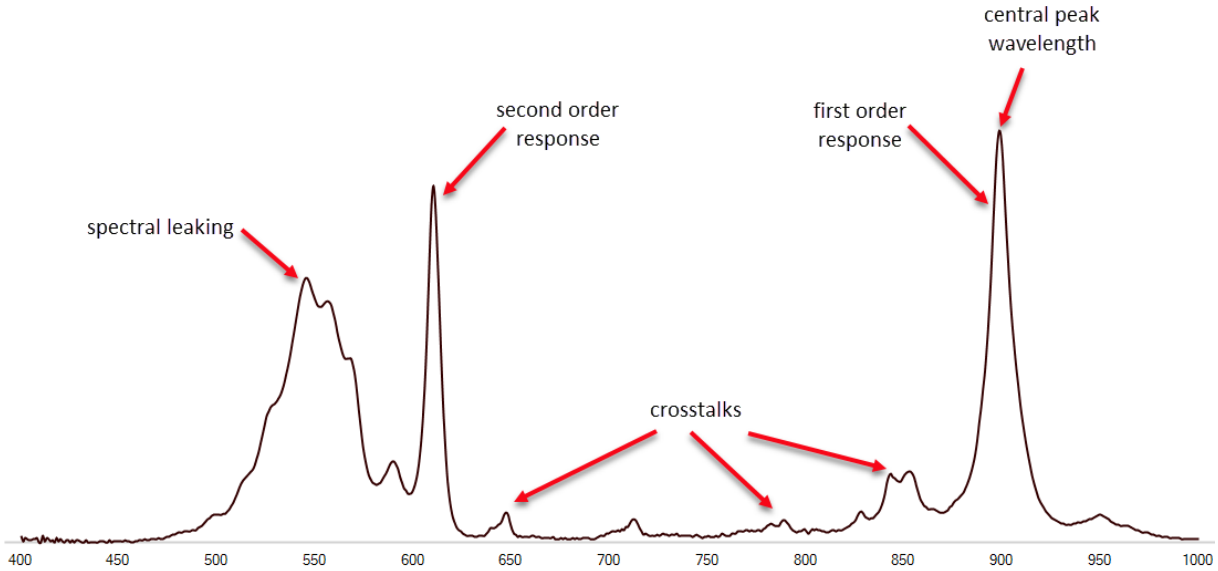


Figure 3-5, visualization of response peaks, crosstalks and spectral leakage from an individual pixel or filter

Crosstalks are often located at the position of other (neighbored) peaks:

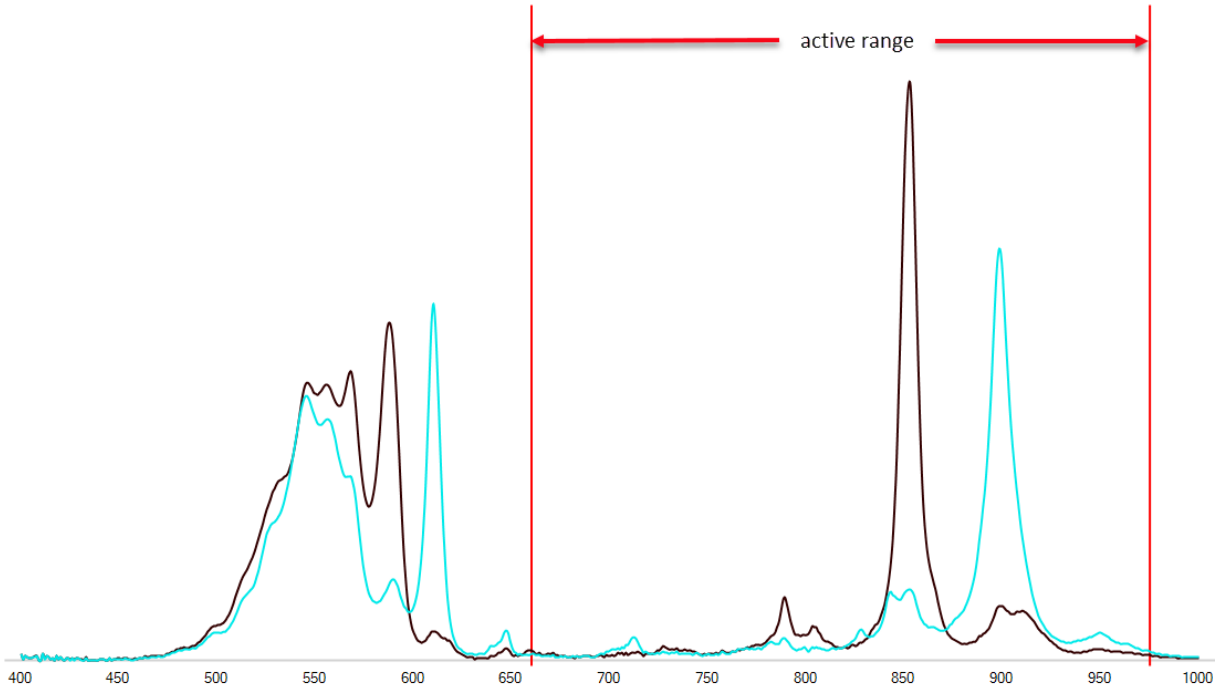


Figure 3-6, visualization of the active range to avoid second harmonics and spectral leakage from two pixels/filters

The sensors are defined to be used with a defined spectral range, the active (wavelength) range. The sensor behavior is not defined outside of this range (due to spectral leakage and additional harmonics).

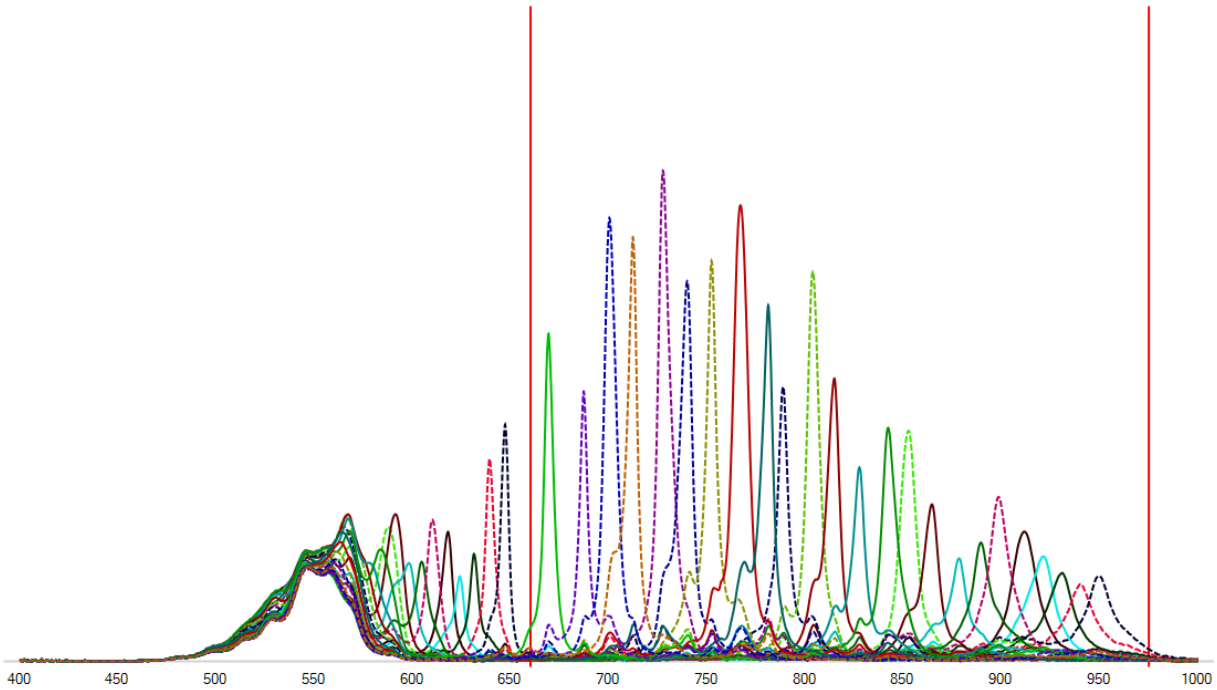


Figure 3-7, visualization of the active range to avoid second harmonics and spectral leakage from a 5x5 camera

Specially designed bandpass filters are built into the xiSpec2 cameras, which limit the light hitting the sensor surface to the active area:

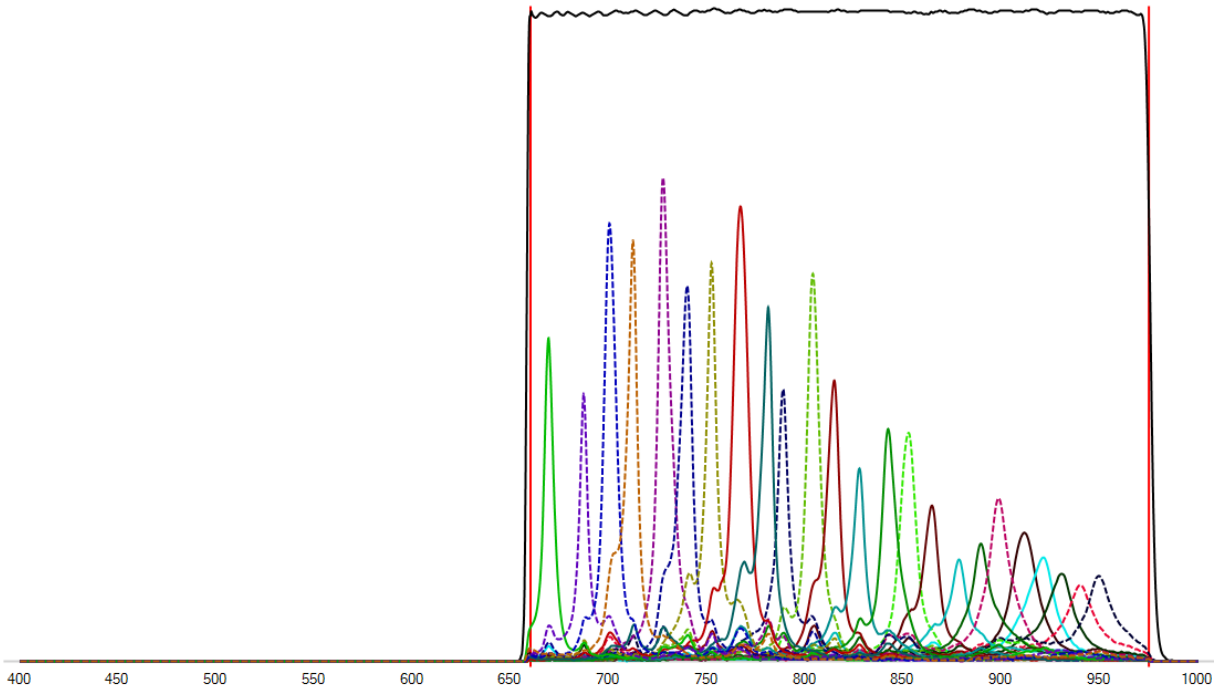


Figure 3-8, Effective response with effect of band-pass filters

The peaks are now almost equidistantly distributed over the active wavelength range. Furthermore, it is now possible to reproduce the position of a band with high accuracy (deviation <= 1%).

3.2. Sensor and camera calibration

All sensors have been calibrated / measured at wafer level. Additionally, every xiSpec2-camera (camera with sensor, filter glass and a standard lens) is spectrally calibrated after production. The camera specific calibration file is part of the scope of delivery.

The calibration data is available in the range of 400 - 1000 nm in 1nm steps.

The bandpass filters used are individually calibrated, the measured transmission curves are part of the calibration data. The filters are often calibrated for a larger wavelength range and with finer subdivisions.

The xiSpec2 cameras are individually measured against standard colorimetric cards. Measurement protocols are included with each camera.

3.3. Camera filter glasses

In xiSpec2 cameras customized bandpass filters are used, which limit the wavelength range of the light reaching the active area of the sensor. The newly developed bandpass filters for the snapshot mosaic sensors are designed to support the widest possible wavelength range while eliminating any second harmonics.

This glass is placed on a layer of silicone, but not glued. Do not use compressed air to clean the camera as this could push dust into the camera. Distance from the lens mounting flange to the sensor surface is 17.526mm +0/– 0.2mm.

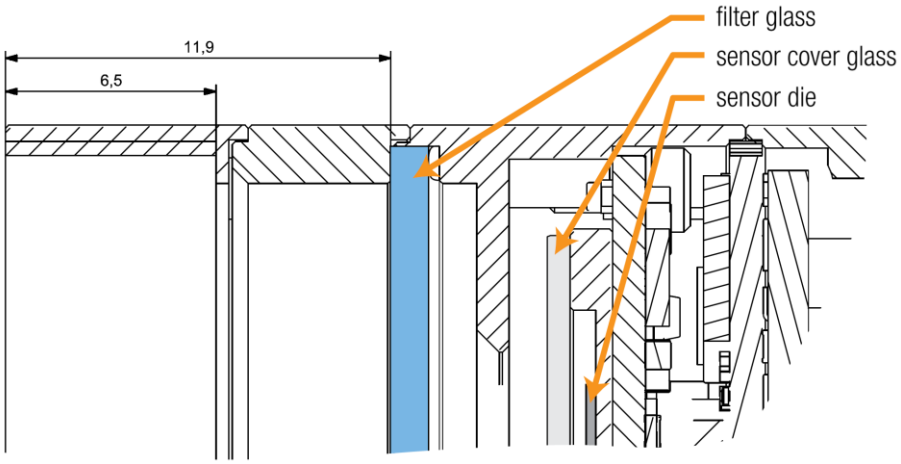


Figure 3-9, position of the customized camera filter glass

3.3.1. Additional filter glasses

Only with the (discontinued) camera models -SM4X4-VIS2 an additional, external filter must be used to limit the wavelength range.

3.4. Lenses

The whole camera setup (camera, camera specific bandpass filters, standard lenses) is calibrated – not only the sensor on wafer level. The calibration results (calibration file and calibration test report) are delivered with the cameras. The lens family which is used for system calibration are optionally available as a part of the starter kits. The recommended aperture is f/2.8.

It is certainly possible to use other lenses, but they cannot be part of the system calibration.

The sensor has a size of 6.0 x 11.3 mm and a diagonal of 12.8mm. The lens should support this image size without vignetting and with a resolution of 5µm or better. Care should be taken that the lenses are as broadband coated as possible, have as little distortion as possible, and that the light exposes the sensor surface as perpendicularly as possible.

The harmonic function (*3.1 Spectral filters and filter response*) demonstrates the dependence of the peak wavelength on the angle of incidence of light. Lenses that ensure an angle below 5° are recommended for use with xiSpec2 cameras. Telecentric lenses on the sensor side are ideal.

3.5. Optimized camera housing

Stray light inside the camera can cause the measurements to be disturbed. Therefore, the inner surfaces of the camera housings were optimized to prevent scattered light. This optimization is achieved by a new special coating on the inside of the camera. The coating is very sensitive and should not be touched.

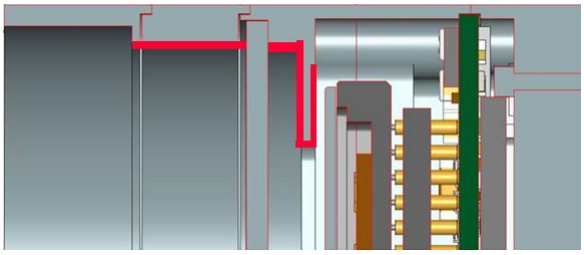


Figure 3-10, specially optimized surfaces in the camera housing, red marked.

4. Filter layouts / camera models / tolerances

xiSpec2 cameras use HSI sensors with different filter layouts. These layouts differ in the basic arrangement of the different filters on the sensor surface.

Basically, two different layouts are supported:

- Snapshot mosaic sensors
- Line scan sensors

4.1. Snapshot mosaic sensors

Three different snapshot-mosaic sensors are available. 4x4 or 5x5 mosaic patterns are repeated continuously on the sensor surface:

- 4x4-Filter-array, active range in the visible light between 460 and 600 nm
- 4x4-Filter-array, active range in the red and near infrared light between 600 and 860 nm
- 5x5-Filter-array, active range in near infrared light between 665 and 960 nm

The spatial resolution is approx. 512x272 pixels (4x4 pattern) or 409x217 pixels (5x5 pattern). The original sensor resolution can be interpolated.

These cameras can be used for real time applications. With the right processing pipeline and speed, each image can be interpreted as a hyperspectral imaging cube immediately.

4.1.1. Camera SM4X4-VIS2 (discontinued)

Wavelength range [nm]	480 – 625 nm
# filters	16
# spectral bands / samples	10
Bands: Peak central wavelengths [nm]	492.3, 503.5, 517.0, 529.7, 554.7, 566.6, 578.8, 589.7, 602.2, 611.8
Calibration file names	CMV2K-SSM4x4-470_620-x.x.x.x.xml

table 4-1, basic sensor specs, SM4X4-VIS2

Peak central wavelength tolerances:

- max. average relative deviation: ±0.8%
- max. single band relative deviation: ±1.0%

The CMV2K-SSM4x4 VIS2 sensor is a CMOSIS CMV2K sensor with 16 filters in a snapshot mosaic layout, active in the visible light. The filter layout is organized in patterns of 4 rows and 4 columns. The 0-based index for the position of the band in the pattern, numbered from left to right, top to bottom. Index positions:

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Figure 4-1, filter index positions, SM4X4-VIS2

The built-in filter glass is only a camera / sensor protection filter. A custom made 480 - 625 nm band pass filter is delivered as an external filter.

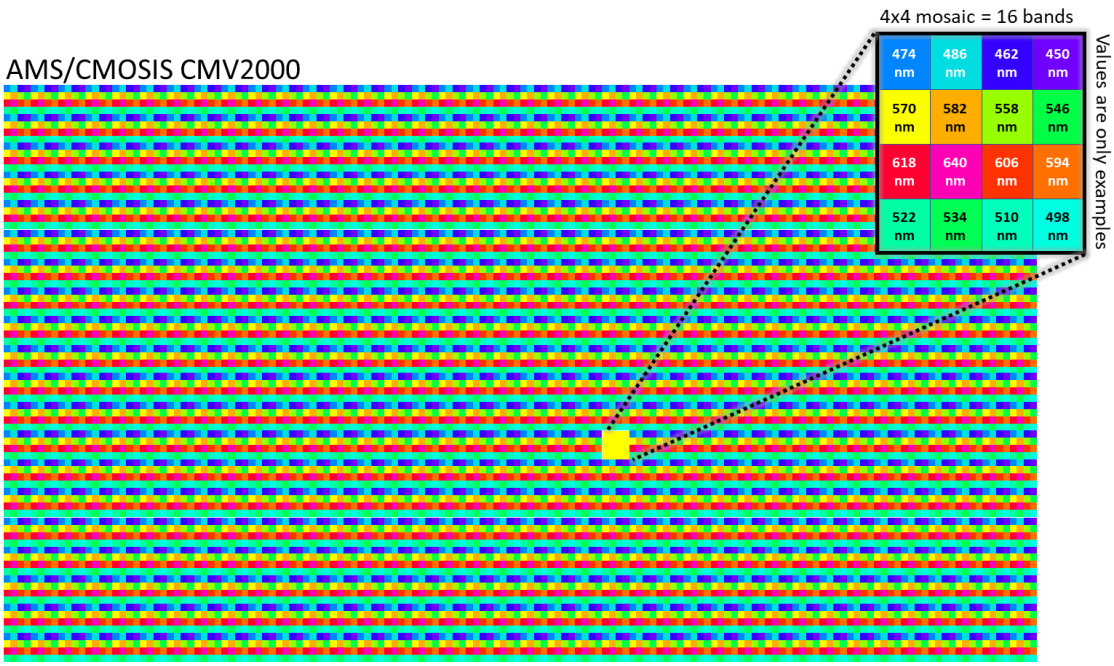


Figure 4-2, filter arrangement at sensor SM4X4-VIS2 (peak wavelength only exemplary)

Example filter response of the SM4x4 VIS2 Sensor:

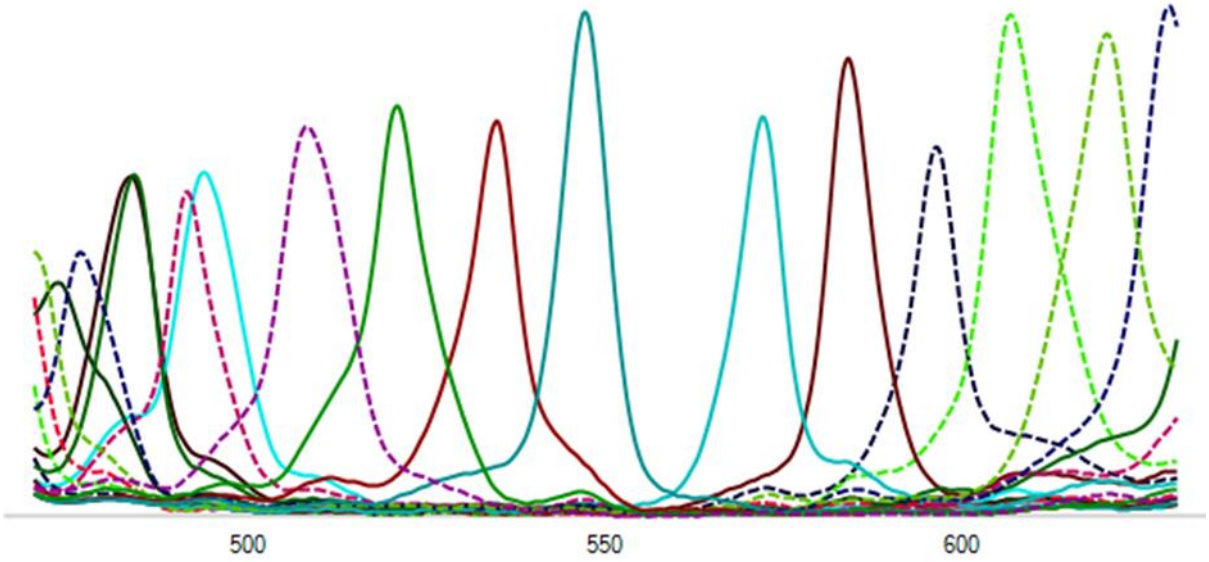


Figure 4-3, example filter response of the SM4X4-VIS2 Sensor

4.1.2. Camera SM4X4-VIS3

Wavelength range [nm]	460 – 600 nm
# filters	16
# spectral bands / samples	16
Bands: Peak central wavelengths [nm]	464.5, 472.8, 480.2, 489.3, 499.0, 508.2, 516.3, 526.1, 534.7, 544.3, 552.3, 561.8, 571.2, 580.5, 588.1, 597.2
Calibration file names	CMV2K-SSM4x4-460_600-x.x.x.x.xml

table 4-2, basic sensor specs, SM4X4-VIS3

Peak central wavelength tolerances:

- max. average relative deviation: ±0.8%
- max. single band relative deviation: ±1.0%

The CMV2K-SSM4x4 VIS3 sensor is a CMOSIS CMV2K sensor with 16 filters in a snapshot mosaic layout, active in the visible light range. The filter layout is organized in patterns of 4 rows and 4 columns. The 0-based index for the position of the band in the pattern, numbered from left to right, top to bottom. Index positions:

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Figure 4-4, filter index positions, SM4X4-VIS3

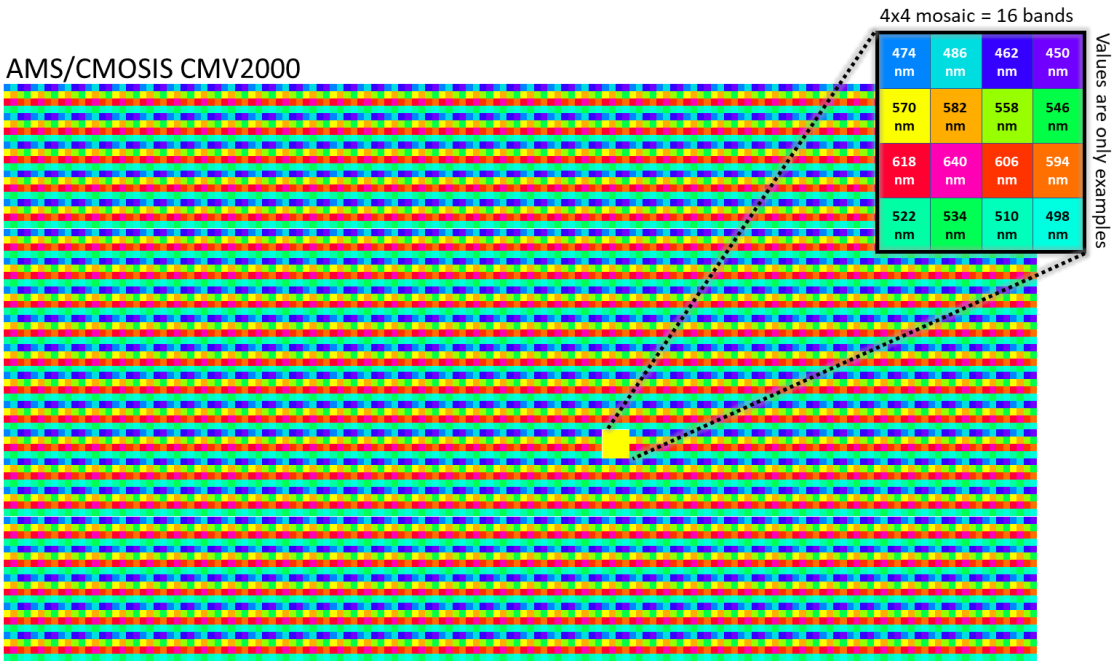


Figure 4-5, filter arrangement at sensor SM4X4-VIS3

Example filter response of the SM4x4 VIS3 Sensor:

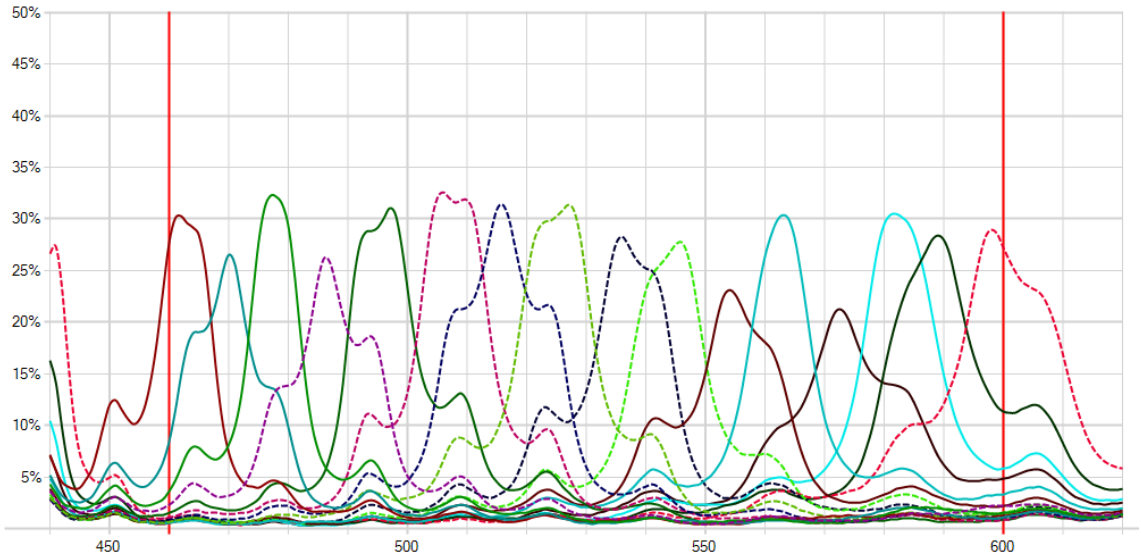


Figure 4-6, example filter response of the SM4X4-VIS3 Sensor

A custom-made band-pass filter is built into the SM4X4-VIS3 camera to limit the wavelength range to the active region of the sensor:

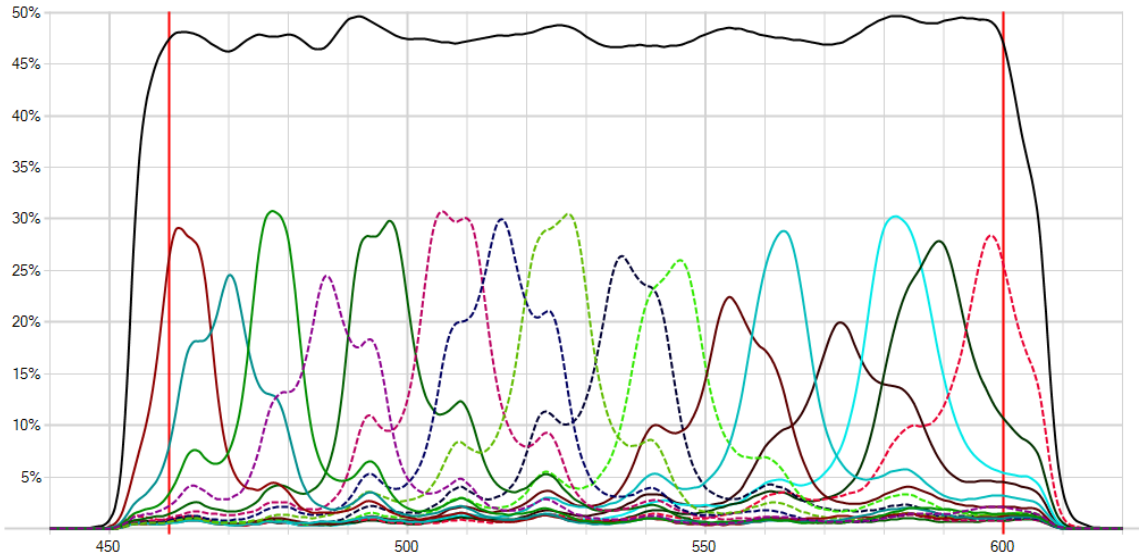


Figure 4-7, Effective response after influence of the band-pass filter (the scale of the filter is 0 - 100%), SM4x4-VIS3

4.1.3. Camera SM4X4-RN2

Wavelength range [nm]	600 – 860 nm
# filters	16
# spectral bands / samples	15
Bands: Peak central wavelengths [nm]	609.0, 625.6, 648.0, 666.3, 683.9, 700.8, 718.9, 736.6, 754.1, 770.1, 786.2, 802.4, 818.3, 833.1, 849.4
Calibration file names	CMV2K-SSM4x4-595_860-x.x.x.x.xml

table 4-3, basic sensor specs, SM4X4-RN2

Peak central wavelength tolerances:

- max. average relative deviation: ±0.8%
- max. single band relative deviation: ±1.0%

The CMV2K-SSM4x4 REDNIR sensor is a CMOSIS CMV2K sensor with 16 filters in a snapshot mosaic layout, active in red and near infrared. The filter layout is organized in patterns of 4 rows and 4 columns. The 0-based index for the position of the band in the pattern, numbered from left to right, top to bottom. Index positions:

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Figure 4-8, filter index positions, SM4X4-RN2

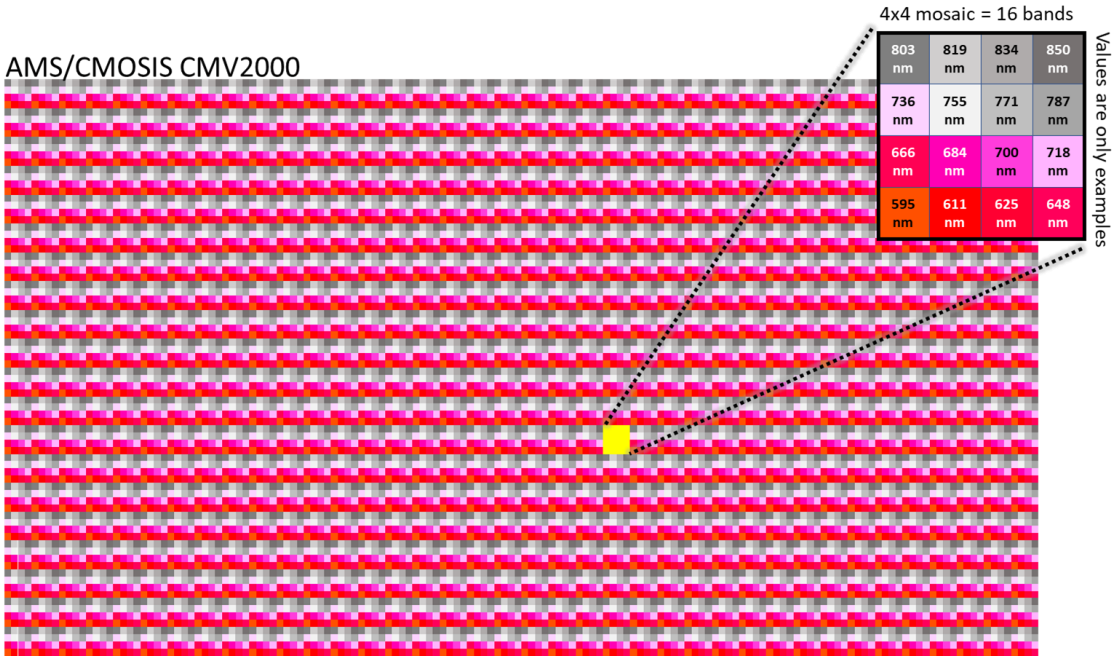


Figure 4-9, filter arrangement at sensor SM4X4-RN2

Example filter response of the SM4x4 RN2 Sensor:

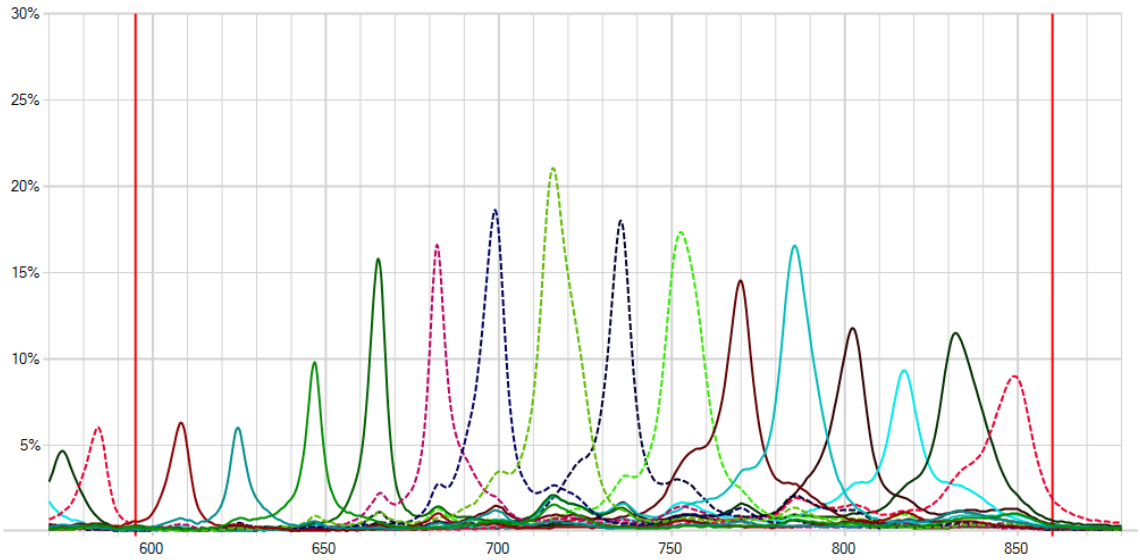


Figure 4-10, example filter response of the SM4X4-RN2 Sensor

A custom-made band-pass filter is built into the SM4X4-RN2 camera to limit the wavelength range to the active region of the sensor:

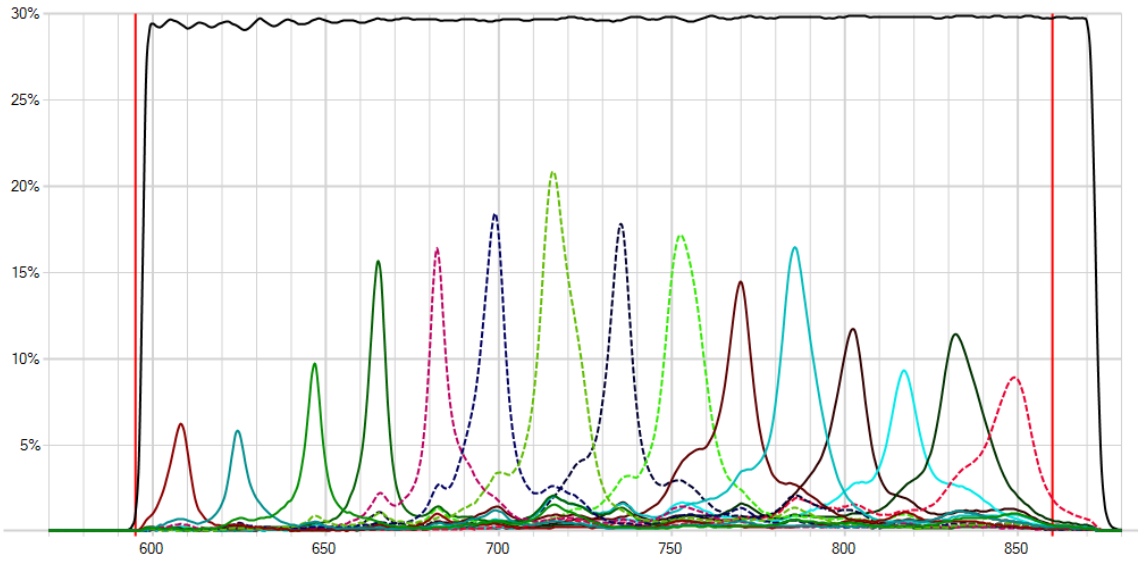


Figure 4-11, Effective response after influence of the band-pass filter (the scale of the filter is 0 - 100%), SM4x4-RN2

4.1.4. Camera SM5X5 NIR2

Wavelength range [nm]	665 – 960 nm
# filters	25
# spectral bands / samples	24
Bands: Peak central wavelengths [nm]	668.7, 686.8, 700.1, 711.6, 728.0, 739.2, 752.3, 767.3, 780.7, 789.4, 804.5, 815.3, 828.3, 843.4, 852.9, 865.1, 879.4, 891.6, 899.8, 912.8, 922.2, 931.9, 942.4, 951.4
Calibration file names	CMV2K-SSM5x5-665_975-x.x.x.x.xml

table 4-4, basic sensor specs, SM5X5-NIR2

Peak central wavelength tolerances:

- max. average relative deviation: ±0.8%
- max. single band relative deviation: ±1.0%

The CMV2K-SSM5x5 NIR2 sensor is a CMOSIS CMV2K sensor with 25 filters in a snapshot mosaic layout, active in the near infrared. The filter layout is organized in patterns of 5 rows and 5 columns. The 0-based index for the position of the band in the pattern, numbered from left to right, top to bottom. Index positions:

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

Figure 4-12, filter index positions, SM5X5-NIR

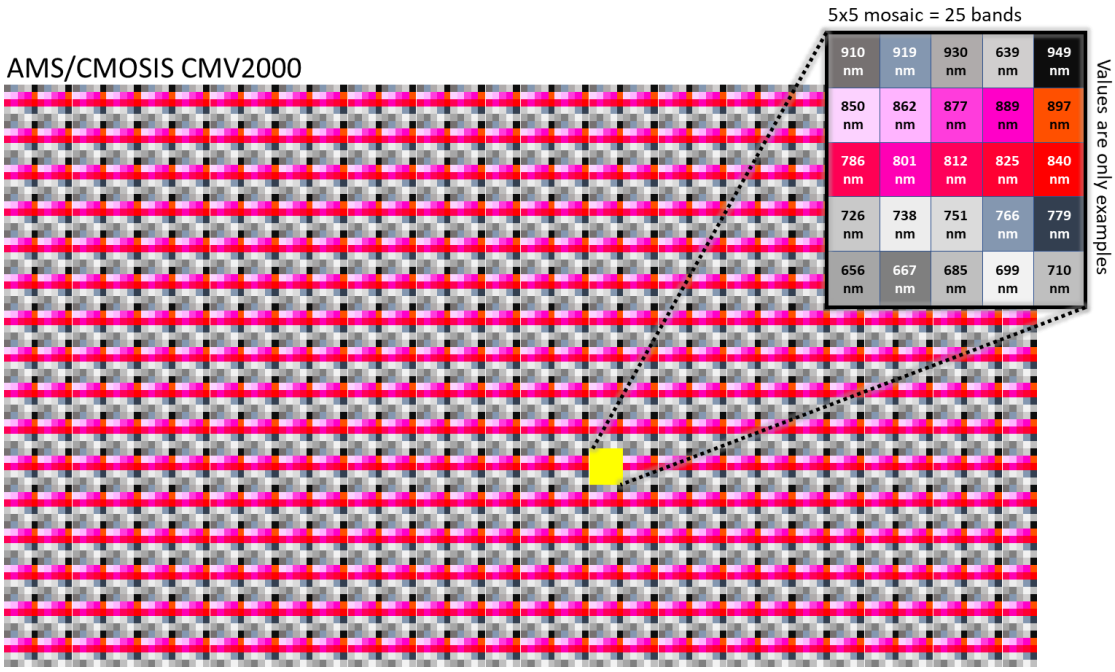


table 4-5 filter arrangement at sensor SM5X5-NIR2

Example filter response of the SM5X5 NIR2 Sensor:

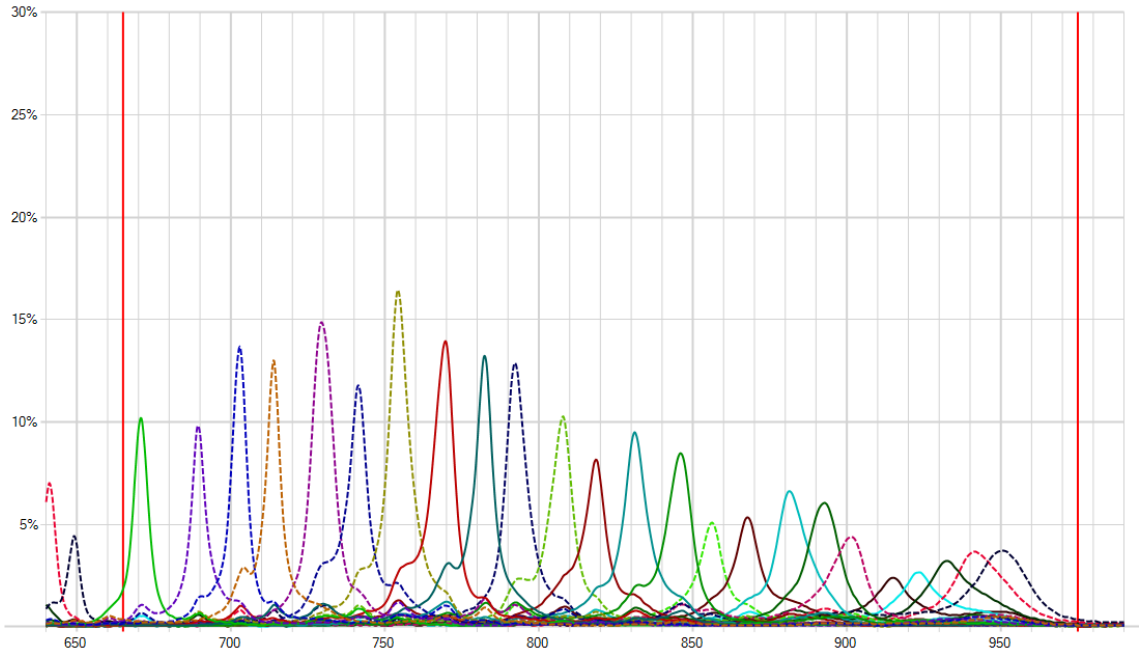


Figure 4-13, example filter response of the SM5X5-NIR2 Sensor

A custom-made band-pass filter is built into the SM5X5-NIR2 camera to limit the wavelength range to the active region of the sensor:

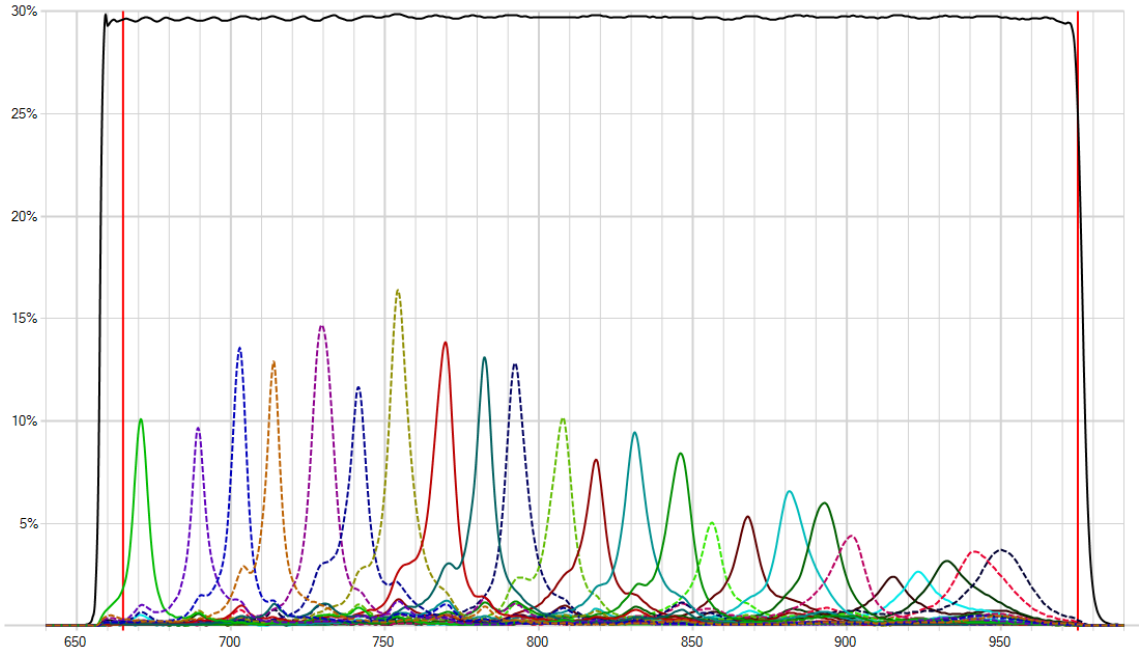


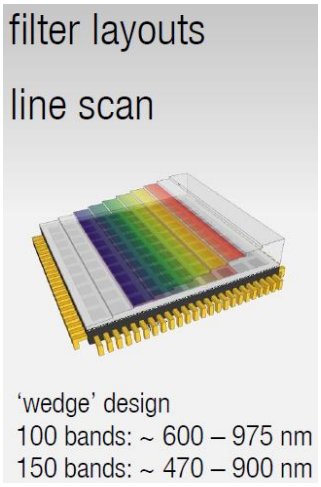
Figure 4-14, Effective response after influence of the band-pass filter (the scale of the filter is 0 - 100%), SM5x5-NIR2

4.2. Line scan sensors

Two line-scan hyperspectral imaging sensors are available:

- LS100 with 100+ spectral bands between 600 and 975 nm in approx. 4nm steps (discontinued sensor)
- LS150 with 150+ spectral bands between 470 and 900 nm in approx. 3mm steps

The line scan filter layout has a wedge design. The *n* filters in the line scan layout are organized in *n* bands of a fixed height over the full width of the active area.



Typically, the width of the active area equals the width of the sensor. The height of the active area equals *n* times the height of the bands in pixels (e.g., 8 pixels). The band at the top of the active area has position index 0. The position index is incremented to the bottom of the active area, with the band at the bottom of the active area has position index *n-1*. The line scan wedge layout is summarized below:

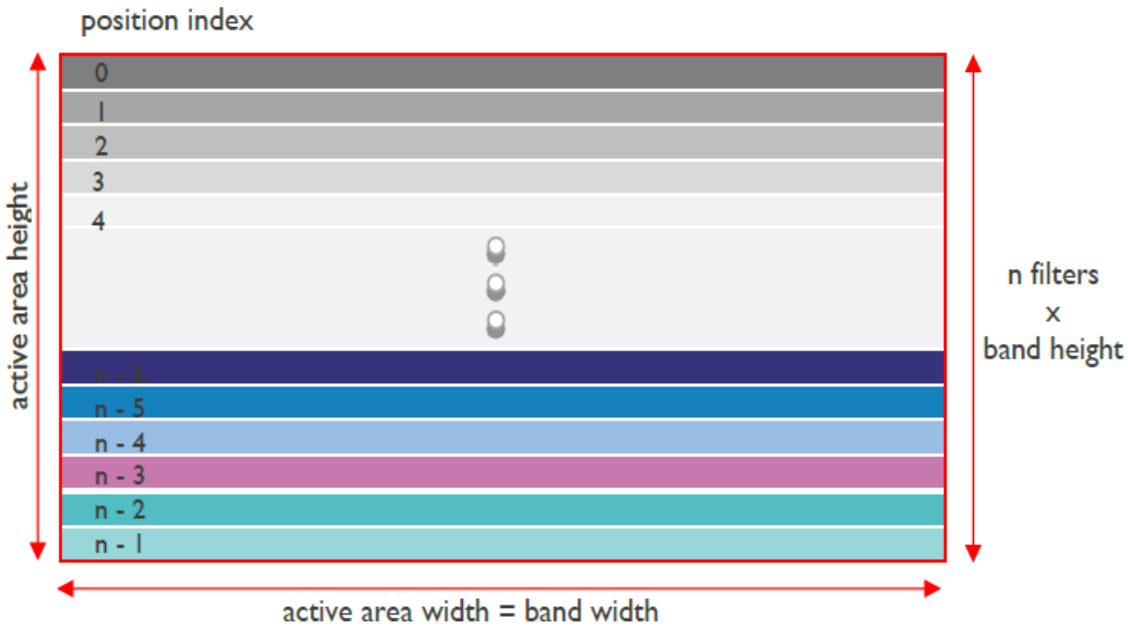


Figure 4-15, basic structure of a line scan sensor (Source: imec)

Because of the organization of the filters in bands over the whole width of the sensor, this filter layout is best suited for line scan applications.

Note that the number of available bands does not necessarily coincide with the actual number of bands on the sensor. This is because some bands are used for production quality checks and future product development.

(Source: imec, "hyperspectral sensors, technology review", V1.1, 2017-10-25)

4.2.1. Camera MQ022HG-IM-LS100 NIR2 (discontinued)

Wavelength range	600 – 975 nm
# spectral bands / samples	100+
Calibration file names	CMV2K-LS100-600_1000-x.x.x.x.xml

table 4-6, basic sensor specs, LS100

The CMV2K LS100 NIR sensor is a CMOSIS CMV2K sensor with 128 filters in a wedge pattern, active in the near infrared (600-975 nm). Each band is 8 rows high, covering 1024 rows of the sensor.

Note: At least 100 of the 128 bands are available for use on the sensor. The remaining bands are used for production quality checks and future product development. Typically, the available bands are band 13 to band 114.

A custom made 600-975 nm band pass filter is built into the XIMEA xiSpec2 camera.

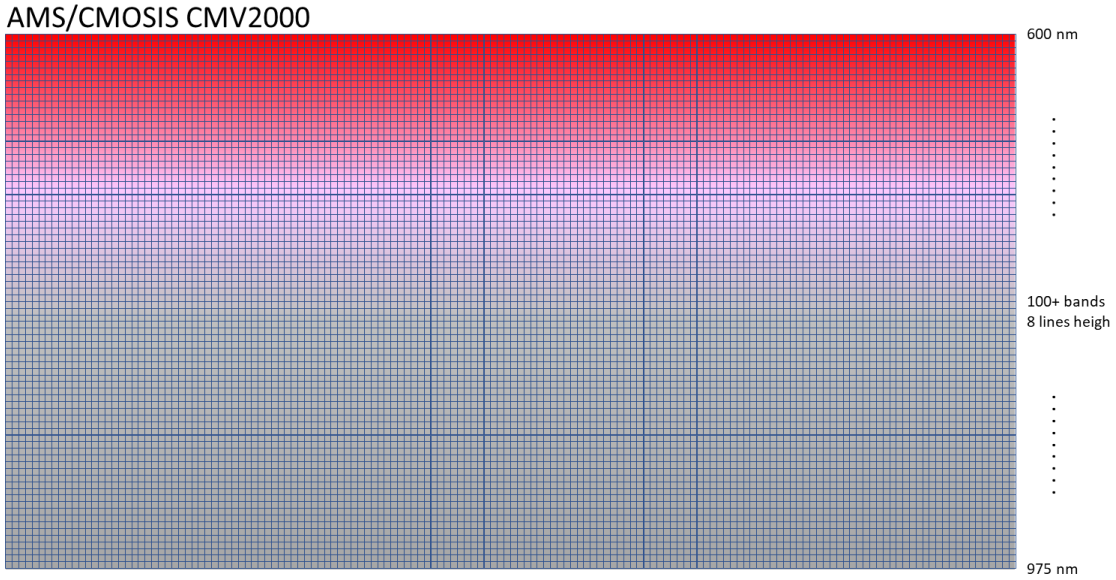


Figure 4-16, filter arrangement at sensor LS100

Example filter responses of the LS100+ NIR sensor in the active range of 600-1000nm:

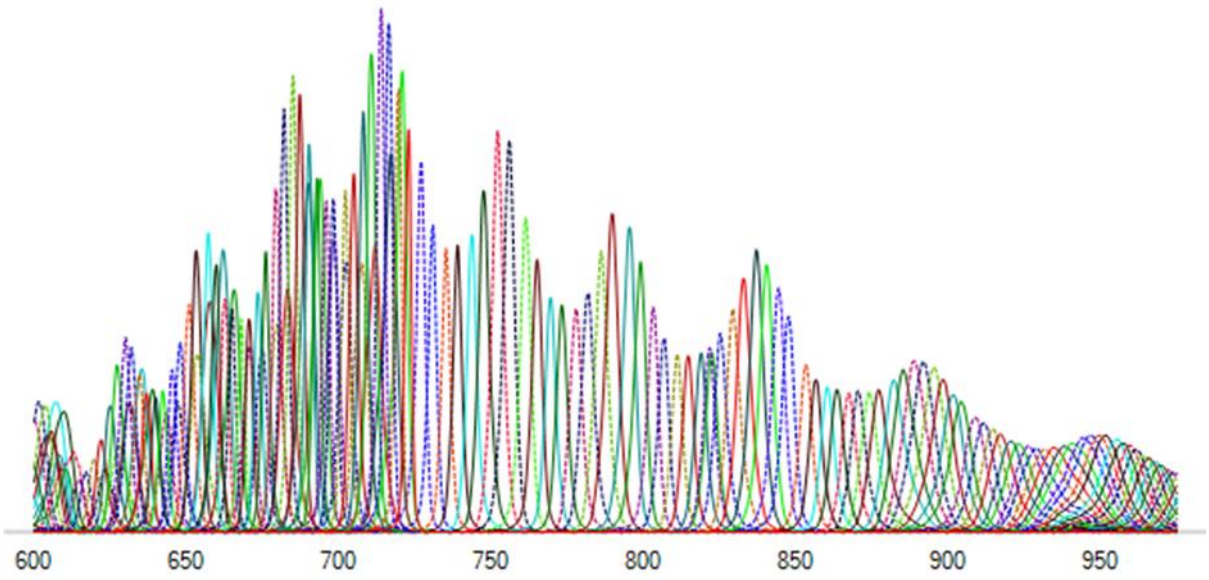


Figure 4-17, example filter response of the LS100 Sensor

Several interference filters in case of the LS100 camera model do have two peak wavelengths in their response curves, the first and second order harmonics.

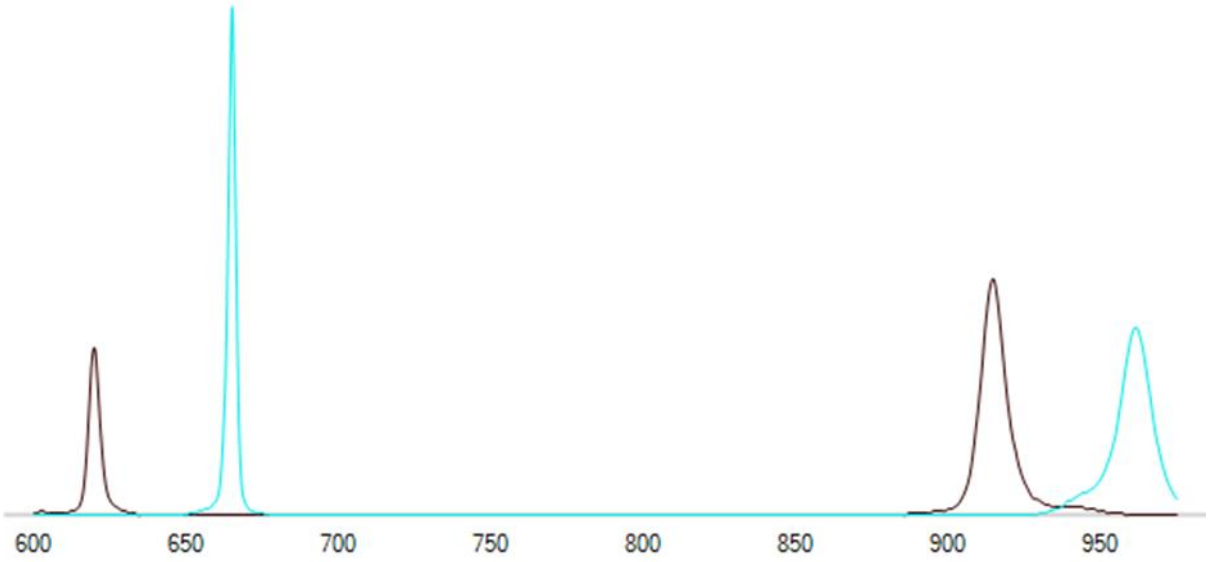


Figure 4-18, example filter response of the LS100 Sensor – bands with first and second harmonics

4.2.1.1. Requirements:

The camera requires a broadband coated lens designed for wavelengths ranges of visible light and near infrared up to 975 nm.

4.2.2. Camera MQ022HG-IM- LS150 NIR

Wavelength range	470 – 900 nm
# spectral bands / samples	150+
Calibration file names	CMV2K-LS150-470_900-x.x.x.x.xml

table 4-7, basic sensor specs, LS150

The CMV2K LS150 VIS-NIR sensor is a CMOSIS CMV2K sensor with 64 filters active in the visual range (470-600 nm) and 128 filters active in the near infrared (600-900 nm). The filters are distributed over two separate active areas. Within each active area, the filters are organized in a wedge pattern in which each band covers 5 rows. The active areas are separated from each other by an empty interface zone of 120 rows. An overview of the filter layout is given in the figure below.

0	VIS
1	VIS
	...
62	VIS
63	VIS
	empty interface zone
64	NIR
65	NIR
	...
190	NIR
191	NIR

Figure 4-19, Principle structure of aLS150 line scan sensor with 2 filter zones (Source: imec)

Note: At least 150 of the 192 bands are available for use on the sensor. The remaining bands are used for production quality checks and future product development.

Note: The pixel response in the empty interface zone is not defined. These pixels are often fully saturated.

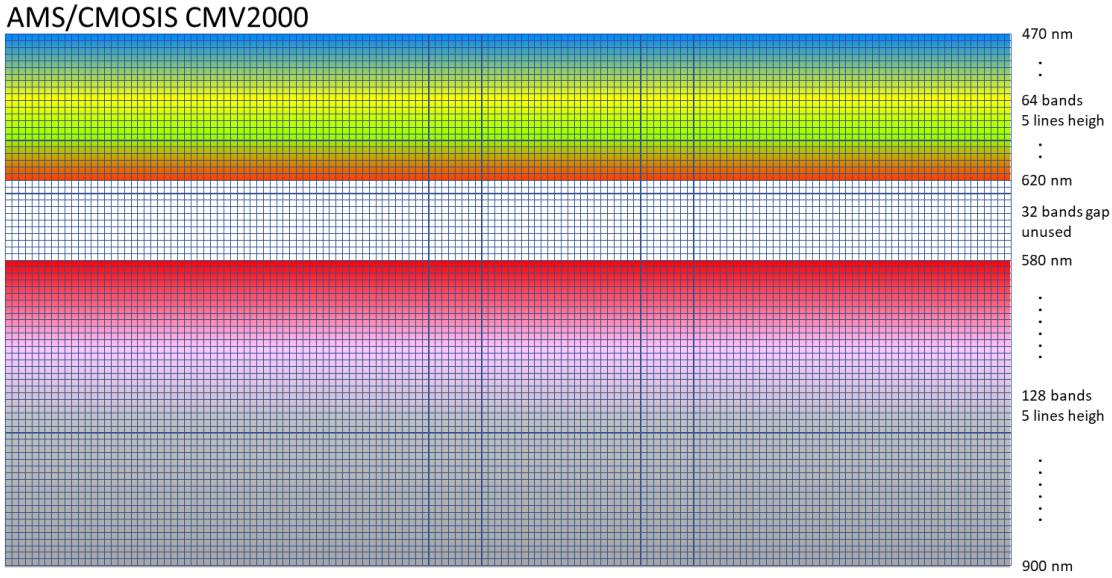


Figure 4-20, filter arrangement at sensor LS150

An example of the responses of the filters is given in the figure below for wavelengths in the range of 400 to 1000 nm. The sensor is designed for an active range of 470-900 nm.

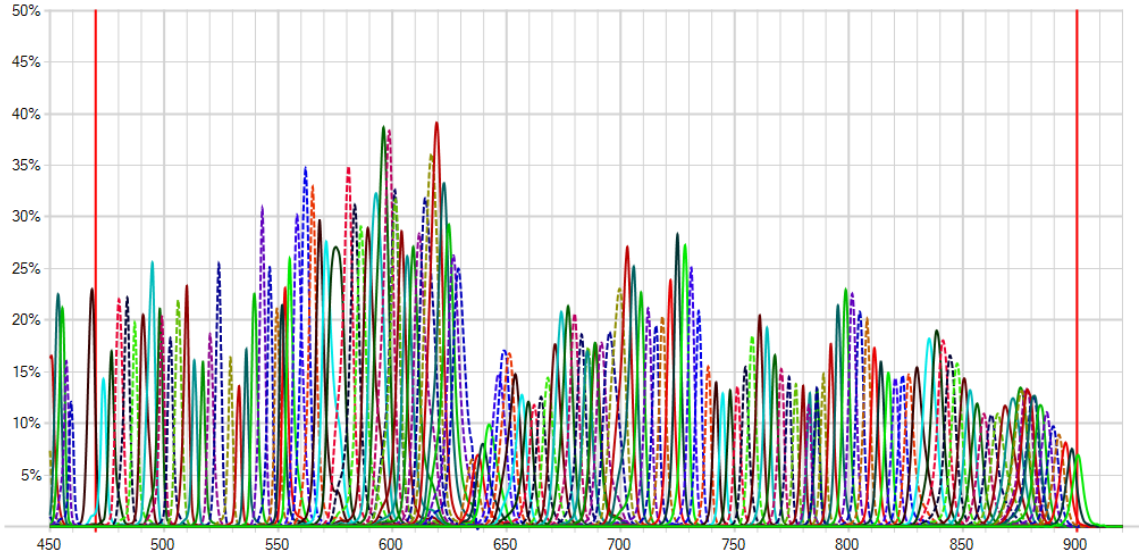


Figure 4-21, example filter response of the LS150 Sensor

A custom-made band-pass filter is built into the SM5X5-NIR2 camera to limit the wavelength range to the active region of the sensor:

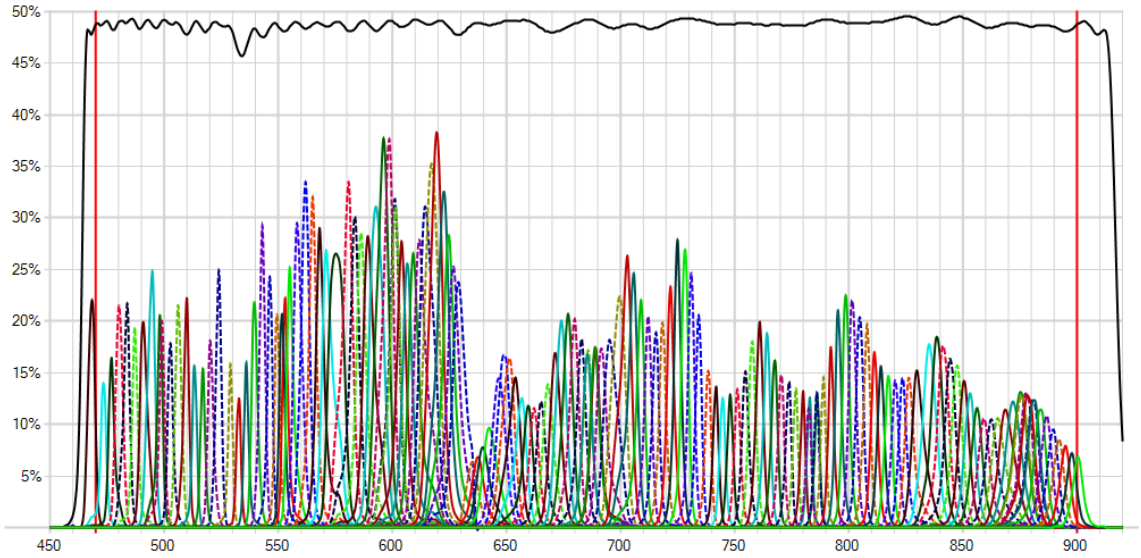


Figure 4-22, Effective response after influence of the band-pass filter (the scale of the filter is 0 - 100%), LS150-VN2

4.2.2.1. Requirements:

The camera requires a broadband coated lens designed for wavelengths ranges of visible light and near infrared up to 900 nm.

4.3. Sensor Defect Specifications

This section enumerates the acceptance criteria, that are used during the calibration and selection, of imec's hyperspectral sensors and xiSpec2 camera systems. The measurements are done with a camera system using an F/8 optical system and on a uniform white target under broadband illumination.

4.3.1. Definitions

4.3.1.1. Pixel Defect

A pixel having a gain that is more than nine standard deviations away from the median gain measured over all pixels on the sensor covered by the same filter and after non-uniformity correction is called a pixel defect.

4.3.1.2. Row/Column Defect

Group of contiguous pixel defects along a single row / column is called a row / column defect. The group size of the pixel defects is dependent on the sensor type and is defined as the cluster threshold (T) in *table 4-8, xiSpec2 sensor defect acceptance criteria*..

4.3.1.3. Cluster Defect

Group of two or more adjacent pixel defects. Size of a cluster is specified in terms of the number of affected pixels, as the cluster threshold (T) in *table 4-8, xiSpec2 sensor defect acceptance criteria*.

4.3.2. Acceptance criteria

Name	Max. number pixel defects	Defective rows/columns	Cluster threshold (T) in pixels	Cluster <= T	Cluster > T
CMV2K-SSM4x4-470_620 CMV2K-SSM4x4-460_600 CMV2K-SSM4x4-595_860	<100	0	4x4	≤5	0
CMV2K-SSM5x5-665_975	<100	0	5x5	≤5	0
CMV2K-LS150-470_900 [1)]	<100	0	10x10	≤5	0

table 4-8, xiSpec2 sensor defect acceptance criteria

Notes: 1) w/o the empty interface zone as described in *4.2.2 Camera MQ022HG-IM- LS150 NIR*

5. xiSpec2 starter kits

5.1. kits - scope of delivery

All xiSpec2 starter kits include the following components:

- tripod bracket
- USB3.0 cable 3m
- Trigger I/O cable 3m
- PH00 screwdriver
- USB3 Type-C OTG adapter
- PELI case with customized foam set
- One of the lenses described below

The starter kits can contain the following optional components:

- Mini tripod
- Lite Diffuse Reflectance target

The following (optional) accessories are available:

Item P/N	Description
XISPEC2-KIT-16	xiSpec2 starter/demo Kit with 16mm lens
XISPEC2-KIT-16-TILE	xiSpec2 starter/demo Kit with 16mm lens and diffuse reflection target
XISPEC2-KIT-16-TILE-TRI	xiSpec2 starter/demo Kit with 16mm lens, diffuse reflection target and mini tripod
XISPEC2-KIT-16-TRI	xiSpec2 starter/demo Kit with 16mm lens and mini tripod
XISPEC2-KIT-25	xiSpec2 starter/demo Kit with 25mm lens
XISPEC2-KIT-25-TILE	xiSpec2 starter/demo Kit with 25mm lens and diffuse reflection target
XISPEC2-KIT-25-TILE-TRI	xiSpec2 starter/demo Kit with 25mm lens, diffuse reflection target and mini tripod
XISPEC2-KIT-25-TRI	xiSpec2 starter/demo Kit with 25mm lens and mini tripod
XISPEC2-KIT-35	xiSpec2 starter/demo Kit with 35mm lens
XISPEC2-KIT-35-TILE	xiSpec2 starter/demo Kit with 35mm lens and diffuse reflection target
XISPEC2-KIT-35-TILE-TRI	xiSpec2 starter/demo Kit with 35mm lens, diffuse reflection target and mini tripod
XISPEC2-KIT-35-TRI	xiSpec2 starter/demo Kit with 35mm lens and mini tripod

table 5-1, xiSpec2 starter kits

5.2. Lens

The lens is an industrial grade NIR corrected, broadband coated lens (425-1000 nm) C-mount lens from Edmund Optics with a filter mount thread (25.5 x 0.5mm) with manually aperture and focus control.

Lenses with focal lengths of 16mm, 25mm and 35mm are available:

- 16mm: FOV horizontal: 38.9°, vertical: 21.2°, WD: 100mm - ∞, weight: 74g
- 25mm: FOV horizontal: 25.5°, vertical: 13.7°, WD: 100mm - ∞, weight: 48g (standard lens)
- 35mm: FOV horizontal: 18.3°, vertical: 9.8°, WD: 165mm - ∞, weight: 75g

Lens selection is a critical component of a hyperspectral imaging project. Longer focal lengths or telecentric designs are recommended to decrease cross-talk between pixels.

Components:

Lens EdmundOptics VIS/NIR lens, C-Mount 35mm fixed focal length with M25.5x0.5mm filter thread (#67-716)
https://www.edmundoptics.com/imaging-lenses/fixed-focal-length-lenses/35mm-c-series-vis-nir-fixed-focal-length-lens/

Lens EdmundOptics VIS/NIR lens, C-Mount 25mm fixed focal length with M25.5x0.5mm filter thread (#67-715)
https://www.edmundoptics.com/imaging-lenses/fixed-focal-length-lenses/25mm-c-series-vis-nir-fixed-focal-length-lens/

Lens EdmundOptics VIS/NIR lens, C-Mount 16mm fixed focal length with M25.5x0.5mm filter thread (#67-714)
https://www.edmundoptics.com/imaging-lenses/fixed-focal-length-lenses/16mm-c-series-vis-nir-fixed-focal-length-lens/

5.3. Lite Diffuse Reflectance target

In order to enable reference images, e.g. to measure the reference light for reflectance calculations and/or for all possible flat-field corrections, it is recommended to use a diffuse reflectance target

5.3.1. Component:

Lite Diffuse Reflectance target, Zenith Lite Target SG-3151-U
http://sphereoptics.de/en/wp-content/uploads/sites/3/2014/03/SphereOptics-Ultralight-Targets-Zenith-Lite.pdf

5.4. PH00 screwdriver

A PH00 screwdriver is required to screw the tripod adapter to the camera.

5.4.1. Component:

Wera 2050 PH00x40mm, 05118019001
https://products.wera.de/en/screwdrivers_series_kraftform_micro_2050_ph_micro.html

5.5. Standard camera accessories:

Several xiQ camera accessories to run the camera are part of the delivery:

- Tripod bracket part MQ-BRACKET-T (xiQ manual chapter 3.12)
- USB3.0 cable 3.0 m part CBL-U3-3M0 (xiQ manual chapter 3.9)
- Trigger I/O cable 3m part CBL-MQSYNC-3M0 (xiQ manual chapter 3.11)
- Mini tripod

Details are described in the xiQ technical manual:
http://www.ximea.com/downloads/usb3/manuals/xiq_technical_manual.pdf

6. USB-Stick

The delivery of each xiSpec2 camera includes a USB stick with the following content:

6.1. Main folder

Here is a readme file and information on how to register with imec's support.

6.2. Folder “Camera files”

The camera-specific calibration file and the acceptance/quality report of the spectral calibration of the camera are stored here.

6.3. Folder “Documentation”

Various documentations are stored in this directory.

6.3.1. xiQ technical manual

The technical manual of our xiQ series (*2.3 Models Overview, sensors and models*). The most recent version is available at:
http://www.ximea.com/downloads/usb3/manuals/xiq_technical_manual.pdf

The installation of our API / SDK is described in chapter 5.3ff.

6.3.2. xiSpec2 technical manual

The technical manual of our xiSpec2 series (this document). The most recent version is available at:
https://www.ximea.com/downloads/usb3/manuals/xispec_technical_manual.pdf

6.3.3. XML schemas

To support your own software development, the USB stick contains the XML schemas of the files that are stored in the file system of the camera, including the sensor calibration. See also: *13.2 XML Schema*.

6.3.4. Calibration files reference (imec)

File Calibration Files Reference Manual.pdf

The most recent file is available in imec’s support portal.

A detailed description is part of this manual: *7 Sensor Calibration data / files*.

6.4. API / SDK / drivers (download):

Our API / drivers etc. are available for download for free:

Most recent beta version: http://www.ximea.com/support/documents/14
Stable version (and LINUX / MacOS files): http://www.ximea.com/support/documents/4

7. Sensor Calibration data / files

The data coding is according to XML 1.0 (Extensible Markup Language (XML) 1.0: http://www.w3.org/TR/xml)

The data encoding is utf-8 / RFC3629: http://www.rfc-editor.org/rfc/rfc3629.txt.

According to W3C XML Schema Definition Language (XSD) the namespace prefix xs id bound to the standard namespace http://www.w3.org/2001/XMLSchema.

All XSD structures are described at https://www.w3.org/TR/xmlschema11-1/ and the datatypes used are described in http://www.w3.org/TR/xmlschema11-2/.

Further documents can be found at https://www.w3.org/TR/?title=xml%20schema.

7.1. XSD XML Schema

The xiSpec2 calibration files use standard XML structures and elements. They correspond to an XML schema and can be serialized / deserialized directly. The XSD structure is listed in Appendix *13.2.2 Calibration files XML Schema V2.0.1*.

The xiSpec2 calibration data remain stable as long as possible.

7.2. Sensor calibration data

The camera specific calibration data are measured by imec (interference filter response data and bandpass-filter transmission data) and calculated (spectral correction) by IMEC. The data format is finally normalized by XIMEA are stored on the USB-stick and in the camera file system. This calibration file format is based on imec's format description V1.9 (April 2021).

XIMEA uses a standard Microsoft .NET 4.7.2 XML-serialization routine to create the XML-file. This may lead to restrictions especially when encoding XML simpleTypes like xs:dateTime. It is explicitly requested not to make any manual changes to the XML calibration files in order not to influence the .NET de-serialization when using the data.

This normalized and stable XML format is described in this chapter.

Please note:

- The XML-type "xs:string (enum)" means that an XML type xs:string is used, but only values from a fixed enumeration are allowed. This enumeration is listed in the description of the element or attribute.
- Some data lists are created in xs:list format. Unfortunately, Microsoft's XML serialization has a restriction. Binding xs:list to XML-elements is unfortunately not supported. Data lists are therefore stored as attributes. Example

```
<response nr_elements="2" values="0.001 0.003" />
```

7.2.1. Root element – sensor_calibration

The xiSpec2 calibration data contains only one (root) element: sensor_calibration.

Attributes:

Name	Description	XML type
version	Version of the calibration file format	xs:int
sensor_id	Four-digit sensor serial number	xs:string
created	Date and time of the sensor calibration	xs:dateTime
modified	Date and time of the last modification of this calibration file	xs:dateTime
software	Reference to the software and version used for the last modification	xs:string

Elements:

Name	Description	XML type
sensor_info	Basic information about the sensor used (AMS/CMOSIS CMV2000)	xs:complexType
filter_info	Characteristics of the Fabry-Perot filters and info about the wavelength range	xs:complexType
system_info	Information about the additional components of the camera (bandpass filter) and spectral correction data	xs:complexType

7.2.2. sensor_info

Basic information about the sensor used.

Attributes:

Name	Description	XML type
version	For compatibility reasons, fixed value = 2	xs:int
sensor_type	Type of the sensor. For xiSpec2 cameras = CMV2K Allowed values: CMV2K CDL640	xs:string (enum)

Elements:

Name	Description	XML type
width_px	Width of the active sensor array, value in pixels	xs:int
height_px	Height of the active sensor array, value in pixels	xs:int
pixel_pitch_um	Pixel size (squared pixels), value in µm	xs:double
bit_depth	Recommended bit depth	xs:int
overall_gain	Recommended overall gain	xs:double
analog_gain	Recommended analog gain	xs:double
digital_gain	The digital gain used during gain calibration	xs:double
full_well_capacity_e	The corresponding full well capacity, in electrons	xs:int
gain_mode	For compatibility reason. Not used for xiSpec2 cameras	xs:string (enum)

Example:

```
<sensor_info version="1">
      <width_px>2048</width_px>
      <height_px>1088</height_px>
      <pixel_pitch_nm>5.5</pixel_pitch_nm>
      <full_well_capacity_e>10443</full_well_capacity_e>
      …
</sensor_info>
```

7.2.3. filter_info

Characteristics of the Fabry-Perot filters and info about the wavelength range

Attributes:

Name	Description	XML type
version	For compatibility reasons, fixed value = 1	xs:int

Elements:

Name	Description	XML type
calibration_info	Information about the sensor calibration measurements	xs:complexType
filter_zones	Information about the filter zones on the sensor	xs:complexType

7.2.4. calibration_info

The characteristics of the filters vary from sensor to sensor. The response curve of each filter on the sensor is measured after production in a monochromator setup using ortho-collimated light at discrete wavelengths approx. from 400 -1000nm in steps of 1nm.

The list of measuring points for sensor calibration is stored here.

Attributes:

Name	Description	XML type
version	For compatibility reasons, fixed value = 5	xs:int

Elements:

Name	Description	XML type
sample_points_nm	list of measuring points for sensor calibration The content is stored in two attributes of this element:	(empty)

name	description	XML-type
nr_elements	Number of elements in the attribute values	xs:int
values	list of measuring points	xs:list of xs:double

Example:

```
<sample_points_nm nr_elements="601" values="399.998 400.999 401.999 … " \>
```

7.2.5. filter_zones

Information about the filter zones on the sensor.

The element filter_zones is just a container of all filter_zone information.

Attributes: None

Elements:

Name	Description	XML type
filter_zone	Information about a filter zone on the sensor surface This element may occur more than once	xs:complexType

7.2.6. filter_zone

Typically, the filters do not cover the entire surface of the sensor. The area on the sensor covered with filters is called the active area. A HSI sensor may have more than one of these active filter zones.

A filter_zone is a collection of info about the sensor area and the bands (Fabry-Perot interference filters) located in that filter_zone.

Attributes:

Name	Description	XML type
version	For compatibility reasons, fixed value = 3	xs:int
layout	Type of the filter arrangement Allowed values: MOSAIC TILED WEDGE	xs:string (enum)
index	0-based index of the filter zone on the sensor	xs:int

Elements:

Name	Description	XML type
filter_area	Area of the sensor on which the filter_zone is located	xs:complexType
pattern_width	Number of different filters along the width of the pattern	xs:int
pattern_height	Number of different filters along the height of the pattern	xs:int
filter_width	Size of the different filters along the width of the pattern, in pixels	xs:int
filter_height	Size of the different filters along the height of the pattern, in pixels	xs:int
spectral_range_start_nm	Lower limit of the wavelength range for which this filter_zone is designed	xs:double
spectral_range_end_nm	Upper limit of the wavelength range for which this filter_zone is designed	xs:double
bands	Container for all interference filters in this zone and their properties	xs:complexType

Notes:

- The width and height of one filter pattern in pixels equals pattern_width x filter_width and pattern_height x filter_height respectively.
- The filter pattern is repeated on the filter area to fill the whole area.

Depending on the sensor model, the filters are organized in one or two filter zones.

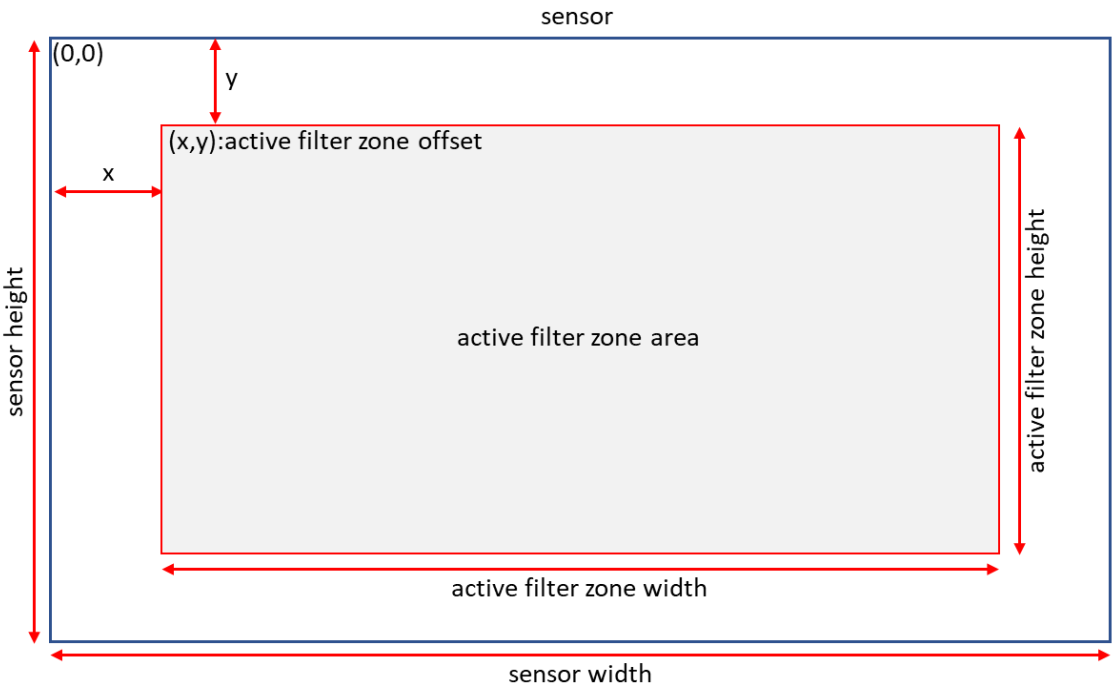


Figure 7-1, active filter area (1 filter zone)

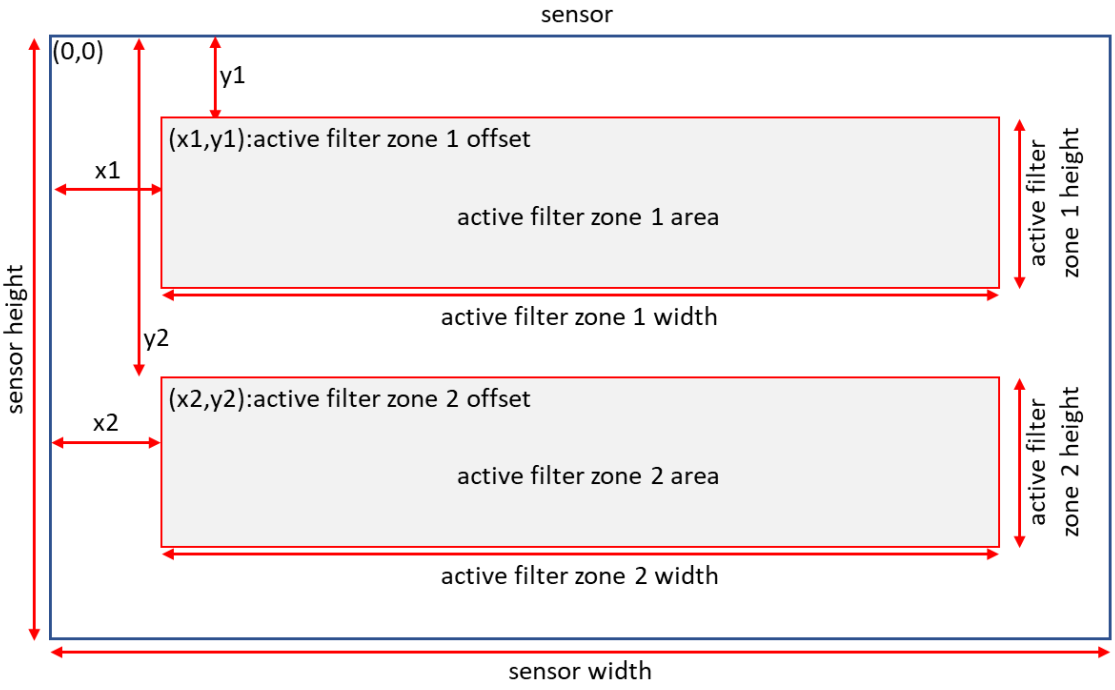


Figure 7-2, active filter areas (2 filter zones)

7.2.7. filter_area

Area of the sensor on which the filter_zone is located.

Attributes:

Name	Description	XML type
Version	For compatibility reasons, fixed value = 0	xs:int

Elements:

Name	Description	XML type
offset_x	column offset of the filter area from the first column of the sensor	xs:int
offset_y	row offset of the filter area from the first row of the sensor	xs:int
width	width of the filter area	xs:int
height	height of the filter area	xs:int

Example:

```
<filter_area version="0">
<offset_x>0</offset_x>
<offset_y>3</offset_y>
<width>2045</width>
<height>1080</height>
</filter_area>
```

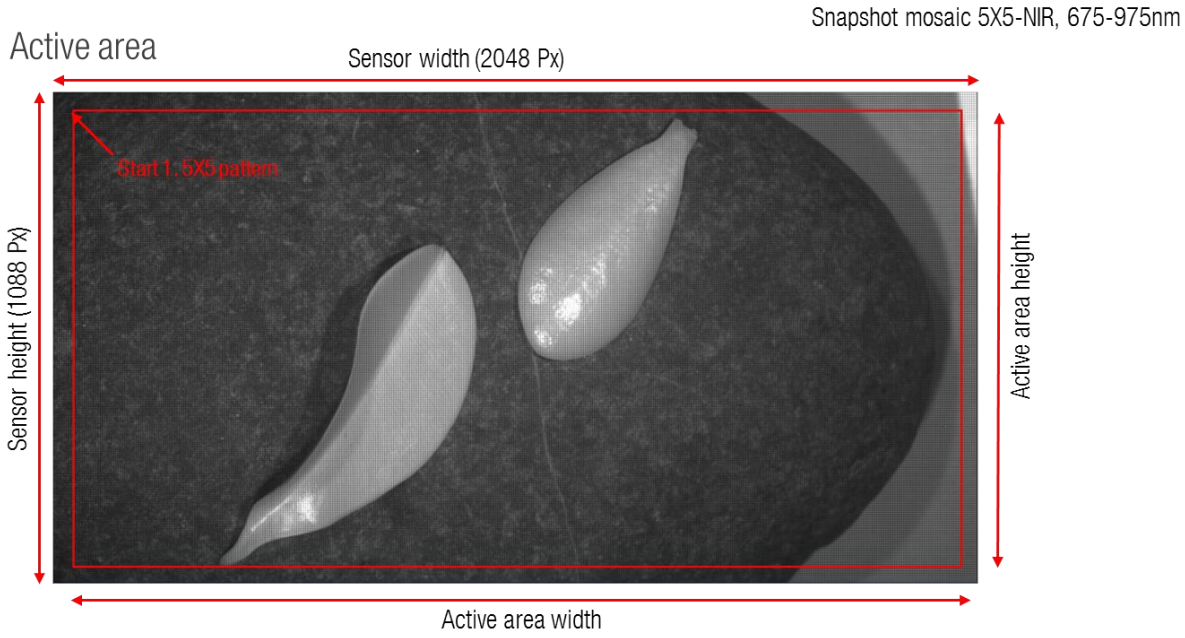


Figure 7-3, example filter area position

7.2.8. bands

Information about the interference filters inside of a filter_zone.

The element bands is just a container of all band information.

Attributes: None

Elements:

Name	Description	XML type
band	Information about interference filter This element may occur more than once	xs:complexType

7.2.9. band

Information about one band (interference filter).

Attributes:

Name	Description	XML type
version	For compatibility reasons, fixed value = 4	xs:int
index	0-based index for the position of the band in the pattern, numbered from left to right, top to bottom	xs:int
selected	Flags if the signal of this band is within specifications (true) or not (false). Usage of non-selected bands will result in wrong spectra	xs:Boolean

Elements:

Name	Description	XML type
peaks	Enumeration of the peaks and their properties in the band's filter responses	xs:complexType
response	list of measured band response / quantum efficiency points for sensor calibration The content is stored in two attributes of this element:	(empty)

name	description	XML-type
nr_elements	Number of elements in the attribute values	xs:int
values	list of response values	xs:list of xs:double

Index positions in case of a SM4X4 sensors:

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Index positions in case of a SM5X5 sensors:

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

The element response contains the spectral composition of a band's response as measured during the calibration. This information depicts the contribution of each wavelength in the illumination to the band's signal.

It is important to note that the response values in the calibration file were measured during sensor manufacture.

Notes:

- The values of the response range from 0 (0%) to 1 (100%) and equals the conversion rate of photons to electrons per wavelength.
- The number of measuring points in the element response corresponds to the number of entries and the response data are the measured values for the corresponding wavelength in the list in calibration_info.sample_point_nm
- There is one band for each position in the pattern and equals filter_info.pattern_width x filter_info.pattern_height

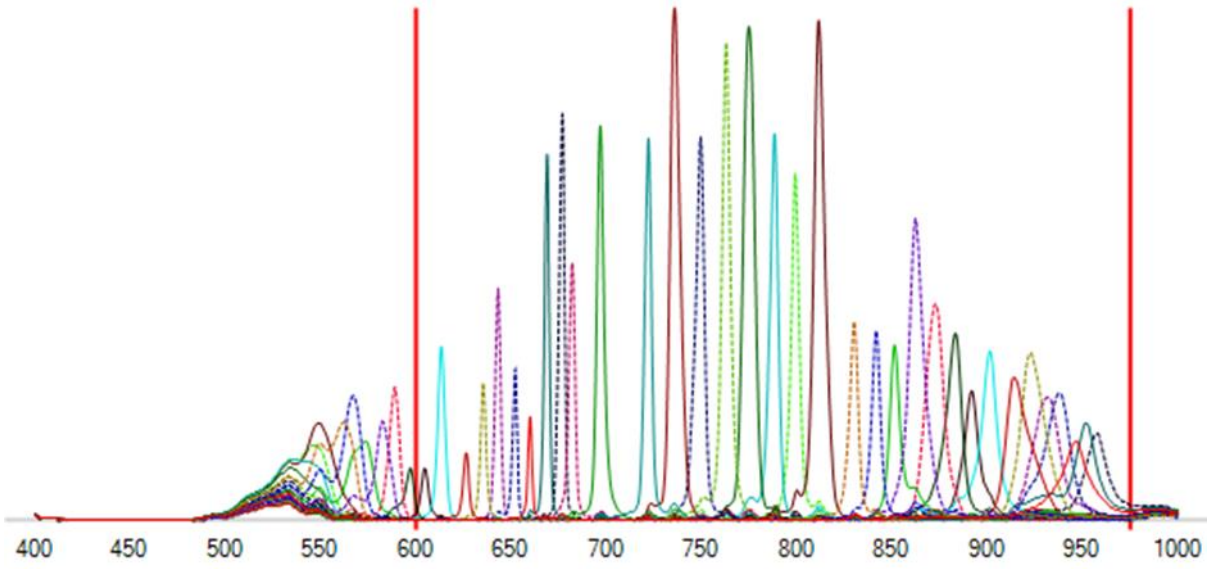


Figure 7-4, example filter response

7.2.10. peaks

Information about the peak(s) in the filter response. The element peaks is just a container of all peak information.

Attributes: None

Elements:

Name	Description	XML type
peak	Information about peaks in a filter response. This element may occur more than once	xs:complexType

7.2.11. peak

Information about one peak in a filter response.

Attributes:

Name	Description	XML type
version	For compatibility reasons, fixed value = 2	xs:int
order	Number of the harmonic order of the peak	xs:int
shape	The filter shape that optimally fits the band responses Allowed values: Fabry-Perot Gaussian Lorentzian	xs:string (enum)

Elements:

Name	Description	XML type
wavelength_nm	The peak wavelength of the ideal Fabry-Perot filter fitted to the measured band response, in nanometers	xs:double
fwhm_nm	The full width of the peak at half the maximum, in nanometers	xs:double
QE	The quantum efficiency of the peak measured at the peak's central wavelength. The values range from 0 (0%) to 1 (100%)	xs:double
contribution	The contribution of the peak's signal to the band's total response. Measured as the area under the fitted peak in [centre ± 1.5xFWHM] relative to the total response	xs:double
fit_error	Goodness of fit to an ideal Fabry-Perot filter shape with given peak wavelength, FWHM and peak QE	xs:double

Example (new calibration file format, SM5X5)

```
<bands>
<band version="4" index="0" selected="true">
      <peaks>
            <peak version="2" order="1" shape="Fabry-Perot">
                  <wavelength_nm>889.9740572143232</wavelength_nm>
                  <fwhm_nm>12.10204081632653</fwhm_nm>
                  <QE>0.0617570698542027</QE>
                  <contribution>0.3458108387442734</contribution>
                  <fit_error>0.003824150970179868</fit_error>
            </peak>
      </peaks>
      <response nr_elements="601" values”0.00316754755, 0.00295955956, …//…..
            0.00313380533” />
</band>
```

7.2.12. system_info

Information about the additional components of the camera (bandpass filter) and spectral correction data.

Attributes:

Name	Description	XML type
version	For compatibility reasons, fixed value = 0	xs:int

Elements:

Name	Description	XML type
optical_components	Enumeration of the optical components in the system	xs:complexType
spectral_correction_info	Information for spectral correction of the raw data	xs:complexType

7.2.13. optical_components

Information about the optical components(s) in the camera. Optical components in xiSpec2 cameras are bandpass filters to support the optimal spectral performance. The element optical_components is just a container of all filter descriptions.

Attributes: None

Elements:

Name	Description	XML type
optical_component	Information about optical components (filters) in a xiSpec2 camera. This element may occur more than once	xs:complexType

7.2.14. optical_component

Information about an optical component / filter in a xiSpec2 camera.

Attributes:

Name	Description	XML type
version	For compatibility reasons, fixed value = 2	xs:int

Elements:

Name	Description	XML type
type	Type of the optical component. Allowed values: shortpass_filter longpass_filter bandpass_filter notch_filter linear_polarizer circular_polarizer source	xs:string (enum)
manufacturer	The manufacturer of the optical component	xs:string
part_id	The unique part identifier as given by the manufacturer	xs:string
tag	Custom tag uniquely identifying the component	xs:string
description	Textual description of the component	xs:string
measurement	Source of the response measurement	xs:string

<table>
<tr><td>transmission_range_start_nm</td><td>Start of the component's transmission range in nanometers, defined as the first 50% transmission point of the targeted spectral range or equal to the first value in sample_points_nm</td><td>xs:double</td></tr>
<tr><td>transmission_range_end_nm</td><td>End of the component's transmission range in nanometres, defined as the last 50% transmission point of the targeted spectral range or equal to the last value in sample_points_nm</td><td>xs:double</td></tr>
<tr><td>measured</td><td>The date on which the optical component was created / measured</td><td>xs:date</td></tr>
<tr><td>sample_points_nm</td><td>Vector to depict the exact wavelengths at which the filters are characterized, in nanometers.
The content is stored in two attributes of this element:
<table>
<tr><th>name</th><th>description</th><th>XML-type</th></tr>
<tr><td>nr_elements</td><td>Number of elements in the attribute values</td><td>xs:int</td></tr>
<tr><td>values</td><td>list of sample points</td><td>xs:list of xs:double</td></tr>
</table></td><td>(empty)</td></tr>
<tr><td>response</td><td>list of measured response (transmission efficiency) for each data point.
The content is stored in two attributes of this element:
<table>
<tr><th>name</th><th>description</th><th>XML-type</th></tr>
<tr><td>nr_elements</td><td>Number of elements in the attribute values</td><td>xs:int</td></tr>
<tr><td>values</td><td>list of response values</td><td>xs:list of xs:double</td></tr>
</table></td><td>(empty)</td></tr>
</table>

Notes:

- The attributes nr_element of the elements sample_points_nm and response must have the same value.
- Both the wavelength range and the number of measurement points used to calibrate the filters may differ from those used to measure the sensor (calibration_info.sample_points_nm).
- The values of response range from 0 (0%) to 1 (100%). The tag's attribute nr_elements describe the number of elements in the list.

The xiSpec2 cameras always contain bandpass filters.

It is important to note that the response values of the sensor (band.response) were measured during sensor manufacture. The effective response values may differ significantly. They are calculated by multiplying the response values by the transmission values of the filters used.

The transmission values of the filters used are stored in the element response.

```
<optical_component version="2">
     . . . . . .
     <type>bandpass_filter</type>
     <response nr_elements="601" values="0 0 .. 0.9356" />
</optical_component>
```

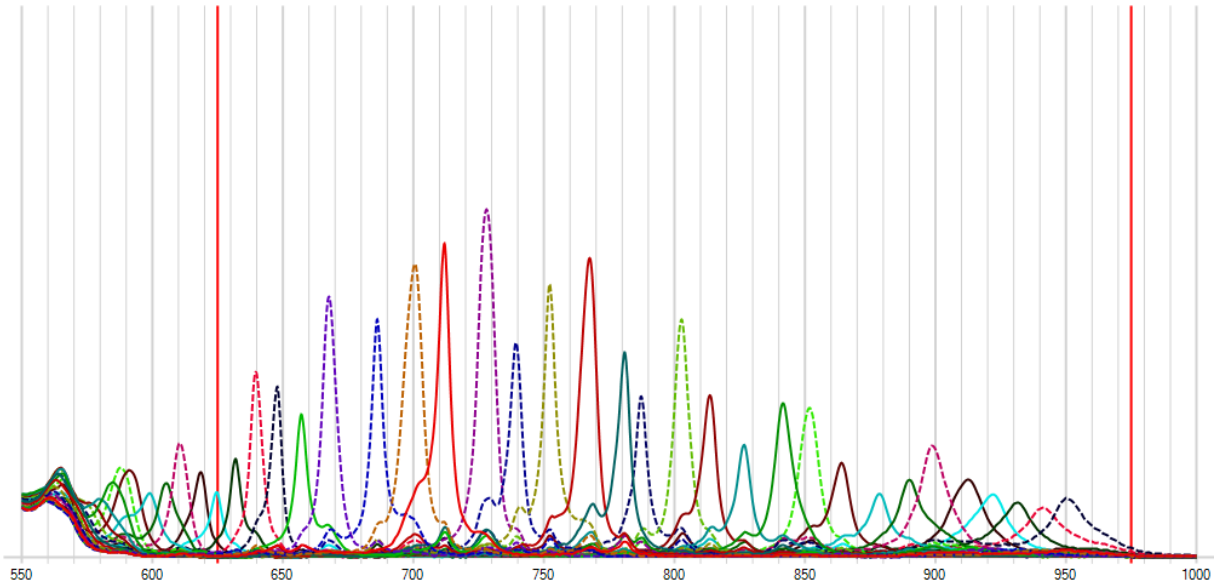


Figure 7-5, response curve, active wavelength range of sensor

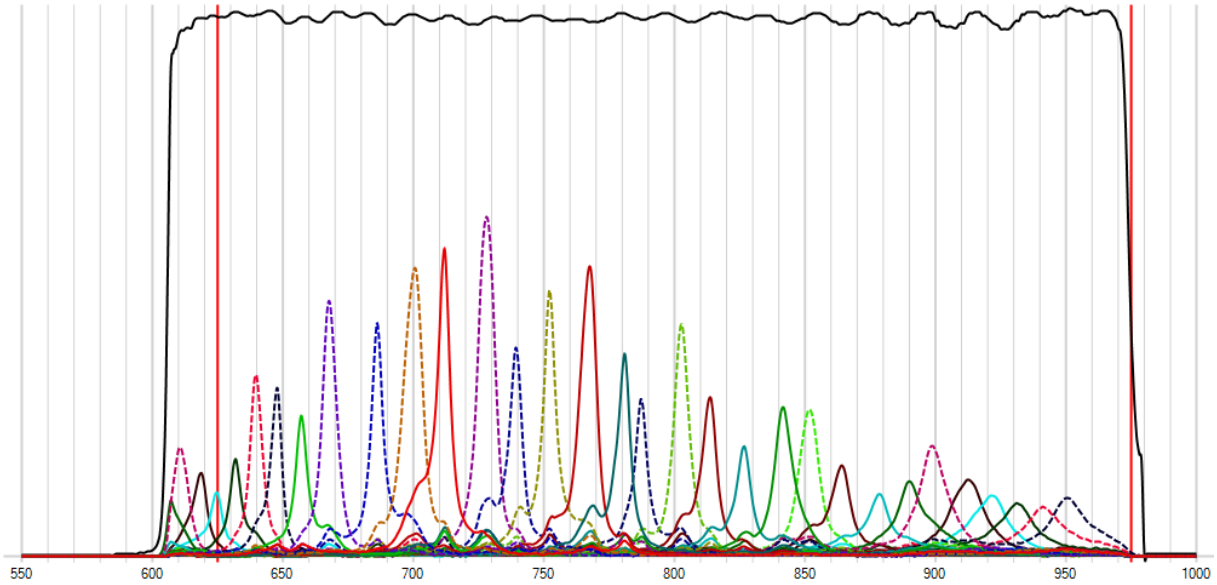


Figure 7-6, response curve under consideration of the camera filter glass

7.2.15. spectral_correction_info

Contains information for spectral correction of the raw data. The spectral correction is done by applying a correction matrix to the measured spectra. This correction matrix is computed by analyzing the response matrix over a specific spectral range and recomposing virtual filters from the response decomposition data.

The spectral correction matrices are enumerated in the child element correction_matrices.

Attributes:

Name	Description	XML type
version	For compatibility reasons, fixed value = 0	xs:int

Elements:

Name	Description	XML type
correction_matrices	Enumeration of correction_matrix	xs:complexType

7.2.16. correction_matrices

Information about the spectral correction data sets. The element correction_matrices is just a container of all correction_matrix entries.

Attributes: None

Elements:

Name	Description	XML type
correction_matrix	Information about spectral correction data sets. This element may occur more than once	xs:complexType

7.2.17. correction_matrix

Information about spectral correction data sets.

Attributes:

Name	Description	XML type
version	For compatibility reasons, fixed value = 6	xs:int
created	Date and time on which the correction matrix was created	xs:dateTime

Elements:

Name	Description	XML type
name	Name of the correction matrix. Any string value is possible.	xs:string
algorithm	the algorithm used to create the matrix. Allowed values: m0 m1	xs:string (enum)
algorithm_version	Version of the algorithm used to generate the correction matrix depicted by two dot-separated numbers. E.g., "1.3". The first number is odd when the correction matrix is computed with ihspt, and even when computed with libs_hsimatlab.	xs:string
type	Type of the spectral correction. Allowed values: reflectance irradiance	xs:string (enum)
minimum_band_energy	Minimum energy of the activated/selected measured band responses, i.e., of those bands used for spectral correction	xs:double
optical_components	Enumeration of all optical components specific for this correction matrix	xs:complexType
virtual_bands	Enumerates the virtual bands that are computed with the spectral correction	xs:complexType

Notes:

- algorithm:
  - m0: correction matrix computed from the estimated system model.
  - m1: correction matrix refined by spectral reference data.
- The generic, i.e., system level optical_components come on top of these listed under system_info. This allows defining both a set of general optical components (e.g., band pass filters) and a set of application specific optical components (e.g., the light spectrum or a refinement of the bandpass filters)
- The minimum_band_energy is computed after applying all optical components to the measured band responses. The energy of a band response is computed as the sum of the final response values over all wavelengths (assuming a 1 nm wavelength sampling).

7.2.18. virtual_bands

Information about the virtual bands which are the results of a spectral correction. The number of (virtual) bands is the effectively usable number of samples in the hyperspectral datacube.

The element virtual_bands is just a container of all virtual_band entries in the correction_matrix.

Attributes: None

Elements:

Name	Description	XML type
virtual_band	Information about a spectral band / sample This element may occur more than once	xs:complexType

7.2.19. virtual_band

Information about a spectral band / sample.

Attributes:

Name	Description	XML type
version	For compatibility reasons, fixed value = 3	xs:int

Elements:

Name	Description	XML type
wavelength_nm	The peak wavelength of the ideal Fabry-Perot filter fitted to the measured band response, in nanometers	xs:double
fwhm_nm	The full width of the peak at half the maximum, in nanometers	xs:double
coefficients	Vector of the spectral correction coefficients. The content is stored in two attributes of this element:	(empty)

Name	description	XML-type
nr_elements	Number of elements in the attribute values	xs:int
values	list of correction coefficients	xs:list of xs:double

Notes:

- Multiplying a measured spectrum with the coefficients results in a spectrally corrected value for the virtual band's wavelength. The attribute nr_elements is the number of bands.
- The matrix in which each row is a virtual band's spectral correction coefficients is also called the sensor's correction matrix. Multiplying this matrix with an irradiance spectrum results in corrected irradiance spectrum.
- The raw spectrum must be sorted on pattern position index and NOT on peak wavelength when multiplying with the correction matrix.

The response curves of the individual bands can contain various crosstalks to other bands, partly also due to further interference harmonics. The correction_matrix_virtual_bands contain correction matrixes to correct the described effects.

If another external filter (usually a long pass or short pass filter) is added, the transmission curve is also taken into account multiplicatively:

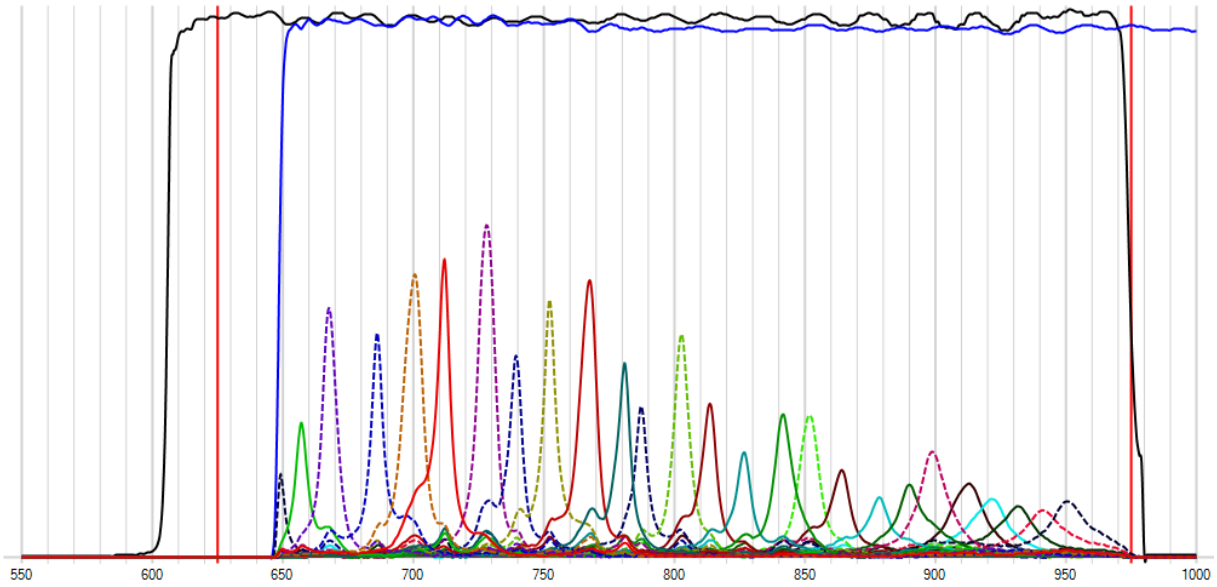


Figure 7-7, response curve under consideration of the camera filter glass and an additional external filter glass

The result of the spectral corrections are the so-called virtual bands, which represent ideal Gauss curves.

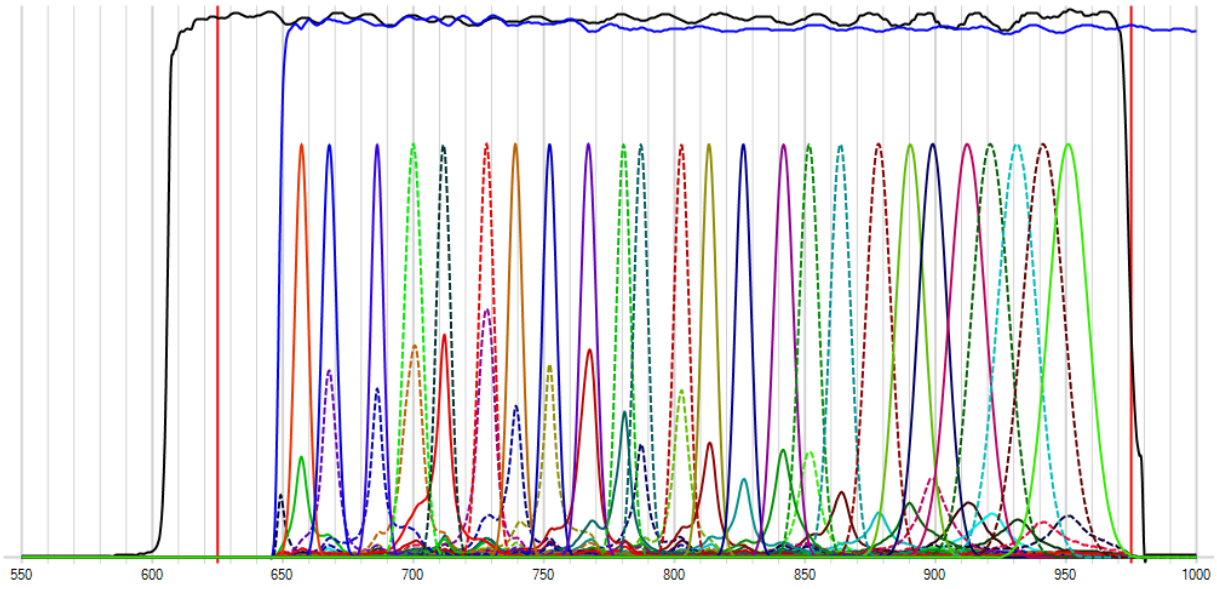


Figure 7-8, calculated virtual bands

Please note details in *9.6 Spectral correction / Correction matrix* .

8. Data Acquisition

8.1. Snapshot mosaic camera – interpretation of the sensor calibration files

All needed info to interpret the calibration file and find the right peak wavelength position in the pattern are described in the calibration file, which is delivered together with the xiSpec2 camera.

8.1.1. RAW image interpretation / snapshot mosaic

Position of the index in the pattern structure:

"The 0-based index for the position of the band in the pattern, numbered from left to right, top to bottom."

Index positions in case of a SM4X4 sensors:

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Index positions in case of a SM5X5 sensors:

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

Pattern interpretation

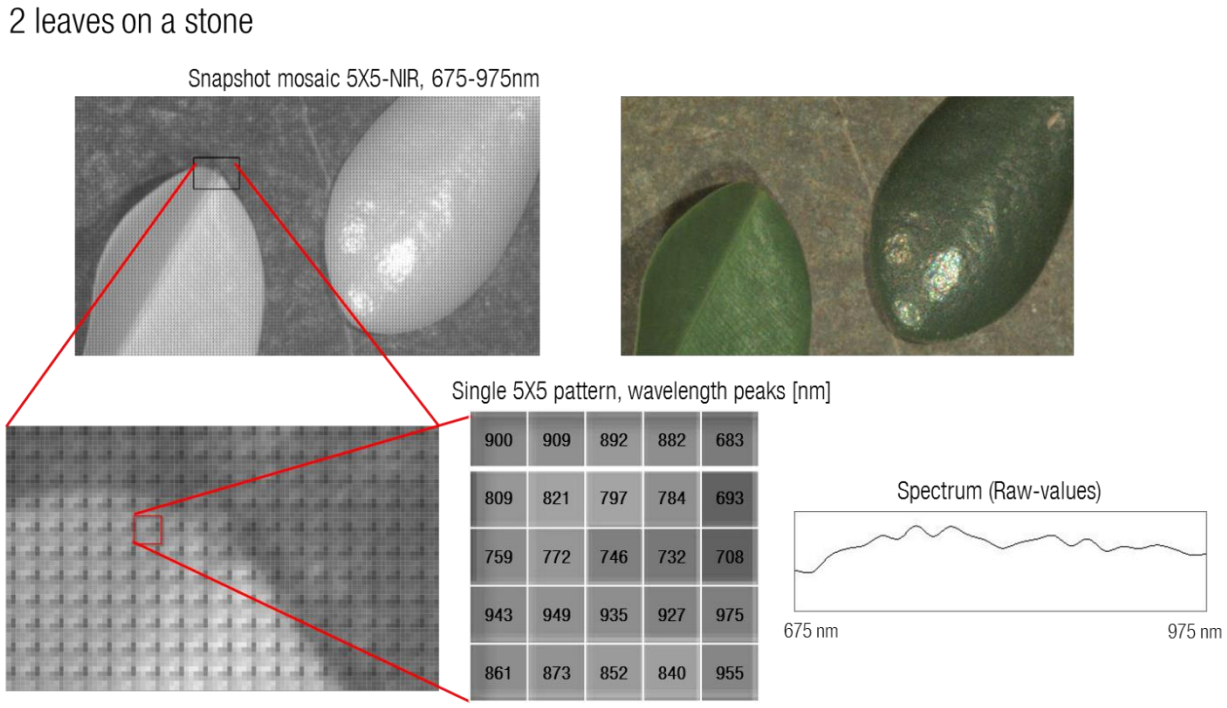


Figure 8-1, visualization and interpretation of a single filter pattern (example SM5X5 with a 675nm long pass filter)

8.1.2. Peak wavelength sort order

The index position only describes the position of the pixel on the sensor surface within the pattern. The index is different from the sort order of the peak wavelengths.

Example (SM4x4-RN2)

801	817	833	848
735	753	769	786
665	682	698	717
802	608	625	647

These wavelengths are rearranged and adjusted during spectral correction as follows, wavelength(index):

608(13), 625(14), 647(15), 665(8), 682(9), 698(10), 717(11), 735(4), 753(5), 769(6), 786(7), 802(0+12), 817(1), 833(2), 848(3)

8.2. Line scan sensors: Data acquisition, data cube and spectrum calculation

Figure 8-2, schematic sequence of data acquisition with movement of sensor / object

The line-scan sensor that is used in xiSpec2 cameras to acquire a hyperspectral representation of an object consists of a conventional sensor with specialized filters added on top. Each filter only transmits a small portion of the complete spectrum reflected by the scene. All this information is then combined with the known motion of the sensor or object to create a hyperspectral representation of the scene: a hypercube.

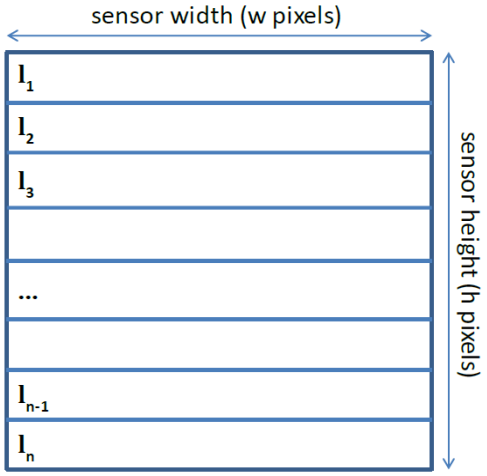


Figure 8-3, line scan (wedge) layout (Source: imec)

A representation of the wedge sensor is given above. It consists of a sensor with a resolution of w x h pixels, with n filter bands processed. Each band has a fixed height in pixels (see above) and a width of w pixels.

As explained, the frequency-specific regions on the sensor are organized in adjacent bands. In order to capture the spectra of every point of an object, we must ensure that every point of the subject passes over each individual band. This is done by moving the object under the sensor or moving the sensor relative to the subject, perpendicular to the orientation of the wedge bands. This is illustrated below:

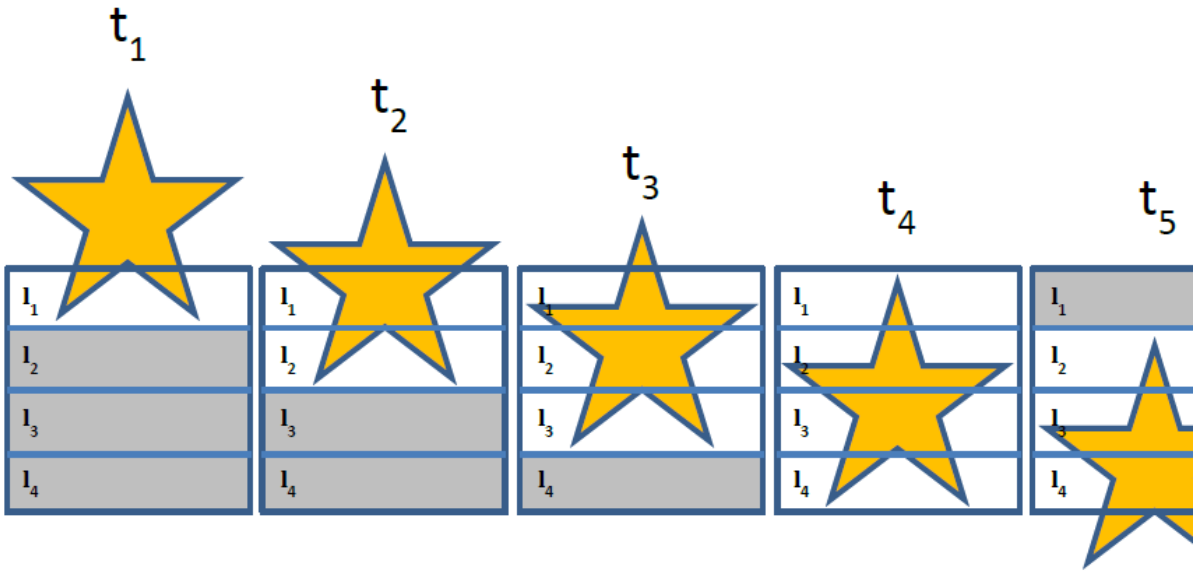


Figure 8-4, line scan sensor – scan phases (Source: imec)

A full scan of an object consists of a start-up phase (t1-t3), a steady-state phase (t4), and a shutdown phase (t5-t7). During the start-up phase and the shutdown phase (t1-t3 and t5-t7, respectively), not all captured data will be used for hypercube construction. This unused data is shown in grey in the figure above. During steady state (see t4), all captured data is used in the hypercube construction. The number of frames in steady state depends on the length of the object.

The construction of a hypercube from these wedge images is done by re-organization of the individual bands, in such a way that all bands of a specific wavelength, taken over consecutive frames, are stitched together, and this for every wavelength. This is illustrated in the figure below. A spectrum of a specific point on the object can be obtained by taking the information for that position over all bands.

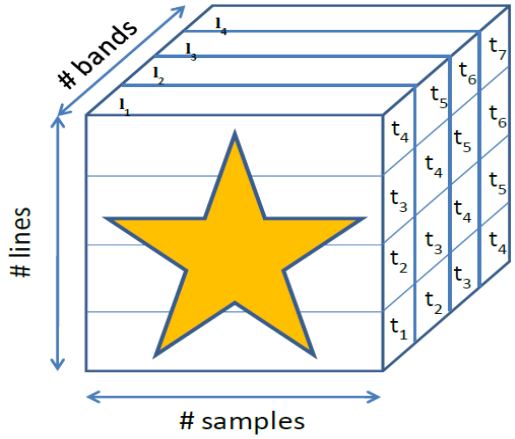
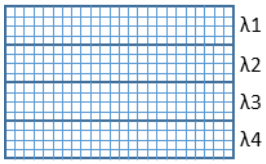


Figure 8-5, line scan sensor – data cube (Source: imec)

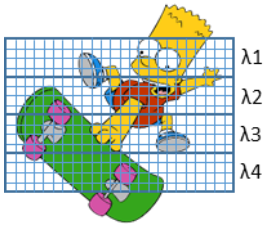
8.2.1.1. Example

We consider as an example a fictitious line scan sensor with 4 spectral bands, each 24 pixels wide and 4 pixels high. The total raw resolution of the sensor is 24 x 16 pixels:



The goal is to capture an object as a hyperspectral data cube that is larger than the camera could capture with an image.

The entire width of the object is captured with an image, but not the height.

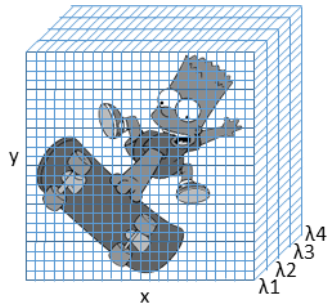


The camera provides an image of a part of the object. This image is always transferred to the computer as a RAW image.

No single transmitted image from a line scan camera can be used to calculate an HSI cube.

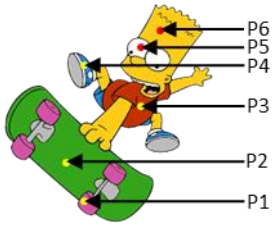
The upper quarter of the image contains only information about the wavelength λ1, the second quarter about the wavelength λ2, etc.

A hyperspectral data cube is a three-dimensional representation in which the spectral information for each pixel (in the x-y-plane) is stored as a third dimension.



We consider 6 different points, which are to be used to represent 6 different materials / spectral signatures:

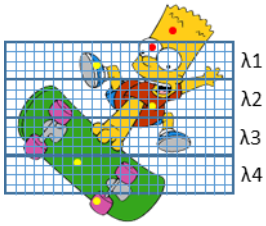
- P1: Wheel
- P2: Skateboard
- P3: T-Shirt
- P4: Shoe
- P5: Eyes
- P6: Skin



The signature vector of a point P is

$$\vec{P} = (\lambda_1, \lambda_2, \lambda_3, \lambda_4)$$

If we now analyze the image



we get the resulting spectral information for the 6 points:

P1 and P6: outside of the RAW image
P2: contains info for spectral position λ4 only
P3: contains info for spectral position λ2 only
P4 and P5: contain info for spectral position λ1 only

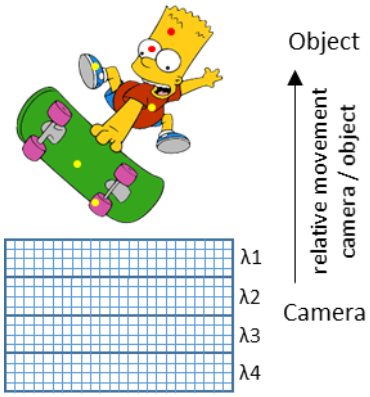
Not a single spectral vector is complete:

$$\vec{P1} = (?, ?, ?, ?)$$
$$\vec{P2} = (?, ?, ?, \lambda_4)$$
$$\vec{P3} = (?, \lambda_2, ?, ?)$$
$$\vec{P4} = (\lambda_1, ?, ?, ?)$$
$$\vec{P5} = (\lambda_1, ?, ?, ?)$$
$$\vec{P6} = (?, ?, ?, ?)$$

To obtain the missing spectral data (other bands) for the object points, these points must also be recorded by the other areas of the sensor that are sensitive in these spectral bands.

To achieve this goal, the object and camera/sensor are moved relative to each other and several images are taken.

This means that all spectral data (bands) can be "collected" for the points of the object.



This process will now be explained in detail. Camera and object are moved relative to each other and several RAW images are recorded and evaluated.

The goal must be to record each point of the object at least once with the 4 different sensor areas (for the 4 peak wavelengths).

If several results for a spectral band are recorded for one pixel, solutions must be provided in the software implementation as to how these different measured values are processed in the calculation of the spectra.

In the following, the mean value is calculated from the various duplicate values.

We take several RAW pictures R1 - Rn.

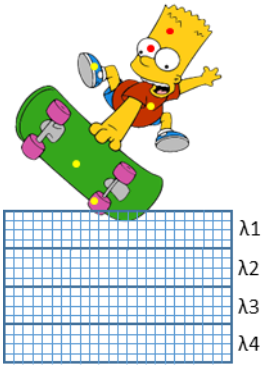
The measured values for one of the points P1-P6 for the bands λ1-λ4 are called:

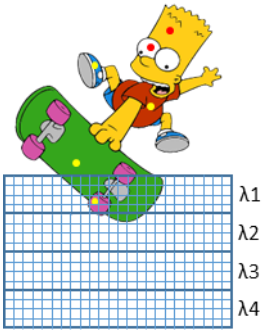
$v_n^{Pm,\lambda k}$

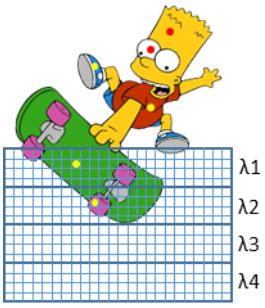
n: picture-#

m: # of the Point (P1-P6)

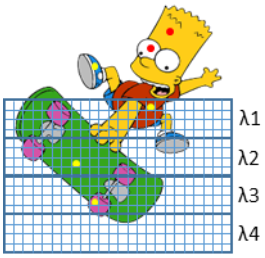
k: # of the band (λ1-λ4)

Picture 1	Measured values	Resulting vectors
	-	$\overrightarrow{P1} = (?,?,?,?)$ $\overrightarrow{P2} = (?,?,?,?)$ $\overrightarrow{P3} = (?,?,?,?)$ $\overrightarrow{P4} = (?,?,?,?)$ $\overrightarrow{P5} = (?,?,?,?)$ $\overrightarrow{P6} = (?,?,?,?)$

Picture 2	Measured values	Resulting vectors
	$v_2^{P1,\lambda 1}$	$\overrightarrow{P1} = (v_2^{P1,\lambda 1},?,?,?)$ $\overrightarrow{P2} = (?,?,?,?)$ $\overrightarrow{P3} = (?,?,?,?)$ $\overrightarrow{P4} = (?,?,?,?)$ $\overrightarrow{P5} = (?,?,?,?)$ $\overrightarrow{P6} = (?,?,?,?)$

Picture 3	Measured values	Resulting vectors
	$v_3^{P1,\lambda 2}$ $v_3^{P2,\lambda 1}$	$\overrightarrow{P1} = (v_2^{P1,\lambda 1}, v_3^{P1,\lambda 2},?,?)$ $\overrightarrow{P2} = (v_3^{P2,\lambda 1},?,?,?)$ $\overrightarrow{P3} = (?,?,?,?)$ $\overrightarrow{P4} = (?,?,?,?)$ $\overrightarrow{P5} = (?,?,?,?)$ $\overrightarrow{P6} = (?,?,?,?)$

Picture 4



Measured values

$v_4^{P1,\lambda 3}$

$v_4^{P2,\lambda 2}$

$v_4^{P3,\lambda 1}$

Resulting vectors

$\overrightarrow{P1} = (v_2^{P1,\lambda 1}, v_3^{P1,\lambda 2}, v_4^{P1,\lambda 3}, ?)$

$\overrightarrow{P2} = (v_3^{P2,\lambda 1}, v_4^{P2,\lambda 2}, ?, ?)$

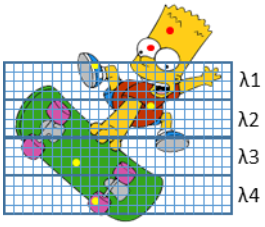
$\overrightarrow{P3} = (v_4^{P3,\lambda 1}, ?, ?, ?)$

$\overrightarrow{P4} = (?, ?, ?, ?)$

$\overrightarrow{P5} = (?, ?, ?, ?)$

$\overrightarrow{P6} = (?, ?, ?, ?)$

Picture 5



Measured values

$v_5^{P1,\lambda 4}$

$v_5^{P2,\lambda 3}$

$v_5^{P3,\lambda 2}$

$v_5^{P4,\lambda 1}$

Resulting vectors

$\overrightarrow{P1} = (v_2^{P1,\lambda 1}, v_3^{P1,\lambda 2}, v_4^{P1,\lambda 3}, v_5^{P1,\lambda 4})$

$\overrightarrow{P2} = (v_3^{P2,\lambda 1}, v_4^{P2,\lambda 2}, v_5^{P2,\lambda 3}, ?)$

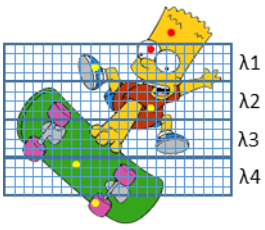
$\overrightarrow{P3} = (v_4^{P3,\lambda 1}, v_5^{P3,\lambda 2}, ?, ?)$

$\overrightarrow{P4} = (v_5^{P4,\lambda 1}, ?, ?, ?)$

$\overrightarrow{P5} = (?, ?, ?, ?)$

$\overrightarrow{P6} = (?, ?, ?, ?)$

Picture 6



Measured values

$v_6^{P2,\lambda 4}$

$v_6^{P3,\lambda 2}$

$v_6^{P4,\lambda 1}$

$v_6^{P5,\lambda 1}$

Resulting vectors

$\overrightarrow{P1} = (v_2^{P1,\lambda 1}, v_3^{P1,\lambda 2}, v_4^{P1,\lambda 3}, v_5^{P1,\lambda 4})$

$\overrightarrow{P2} = (v_3^{P2,\lambda 1}, v_4^{P2,\lambda 2}, v_5^{P2,\lambda 3}, v_6^{P2,\lambda 4})$

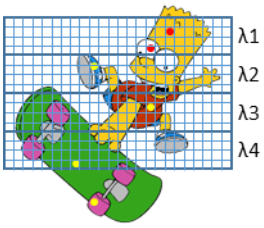
$\overrightarrow{P3} = (v_4^{P3,\lambda 1}, \frac{v_5^{P3,\lambda 2} + v_6^{P3,\lambda 2}}{2}, ?, ?)$

$\overrightarrow{P4} = (\frac{v_5^{P4,\lambda 1} + v_6^{P4,\lambda 1}}{2}, ?, ?, ?)$

$\overrightarrow{P5} = (v_6^{P5,\lambda 1}, ?, ?, ?)$

$\overrightarrow{P6} = (?, ?, ?, ?)$

Picture 7



Measured values

$v_7^{P2,\lambda4}$

$v_7^{P3,\lambda3}$

$v_7^{P4,\lambda2}$

$v_7^{P5,\lambda1}$

$v_7^{P6,\lambda1}$

Resulting vectors

$$\overrightarrow{P1} = (v_2^{P1,\lambda1}, v_3^{P1,\lambda2}, v_4^{P1,\lambda3}, v_5^{P1,\lambda4})$$

$$\overrightarrow{P2} = (v_3^{P2,\lambda1}, v_4^{P2,\lambda2}, v_5^{P2,\lambda3}, \frac{v_6^{P2,\lambda4} + v_7^{P2,\lambda4}}{2})$$

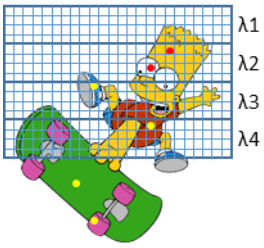
$$\overrightarrow{P3} = (v_4^{P3,\lambda1}, \frac{v_5^{P3,\lambda2} + v_6^{P3,\lambda2}}{2}, v_7^{P3,\lambda3}, ?)$$

$$\overrightarrow{P4} = (\frac{v_5^{P4,\lambda1} + v_6^{P4,\lambda1}}{2}, v_7^{P4,\lambda2}, ?, ?)$$

$$\overrightarrow{P5} = (\frac{v_6^{P5,\lambda1} + v_7^{P5,\lambda1}}{2}, ?, ?, ?)$$

$$\overrightarrow{P6} = (v_7^{P6,\lambda1}, ?, ?, ?)$$

Picture 8



Measured values

$v_8^{P3,\lambda4}$

$v_8^{P4,\lambda3}$

$v_8^{P5,\lambda2}$

$v_8^{P6,\lambda2}$

Resulting vectors

$$\overrightarrow{P1} = (v_2^{P1,\lambda1}, v_3^{P1,\lambda2}, v_4^{P1,\lambda3}, v_5^{P1,\lambda4})$$

$$\overrightarrow{P2} = (v_3^{P2,\lambda1}, v_4^{P2,\lambda2}, v_5^{P2,\lambda3}, \frac{v_6^{P2,\lambda4} + v_7^{P2,\lambda4}}{2})$$

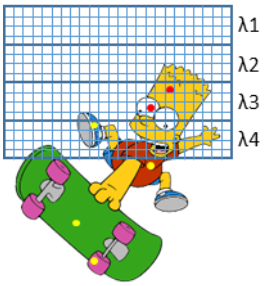
$$\overrightarrow{P3} = (v_4^{P3,\lambda1}, \frac{v_5^{P3,\lambda2} + v_6^{P3,\lambda2}}{2}, v_7^{P3,\lambda3}, v_8^{P3,\lambda4})$$

$$\overrightarrow{P4} = (\frac{v_5^{P4,\lambda1} + v_6^{P4,\lambda1}}{2}, v_7^{P4,\lambda2}, v_8^{P4,\lambda3}, ?)$$

$$\overrightarrow{P5} = (\frac{v_6^{P5,\lambda1} + v_7^{P5,\lambda1}}{2}, v_8^{P5,\lambda2}, ?, ?)$$

$$\overrightarrow{P6} = (v_7^{P6,\lambda1}, v_8^{P6,\lambda2}, ?, ?)$$

Picture 9



Measured values

$v_9^{P4,\lambda4}$

$v_9^{P5,\lambda3}$

$v_9^{P6,\lambda3}$

Resulting vectors

$$\overrightarrow{P1} = (v_2^{P1,\lambda1}, v_3^{P1,\lambda2}, v_4^{P1,\lambda3}, v_5^{P1,\lambda4})$$

$$\overrightarrow{P2} = (v_3^{P2,\lambda1}, v_4^{P2,\lambda2}, v_5^{P2,\lambda3}, \frac{v_6^{P2,\lambda4} + v_7^{P2,\lambda4}}{2})$$

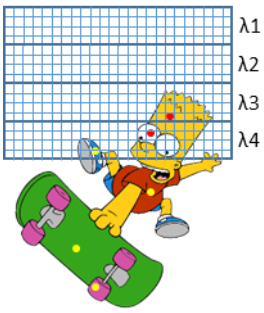
$$\overrightarrow{P3} = (v_4^{P3,\lambda1}, \frac{v_5^{P3,\lambda2} + v_6^{P3,\lambda2}}{2}, v_7^{P3,\lambda3}, v_8^{P3,\lambda4})$$

$$\overrightarrow{P4} = (\frac{v_5^{P4,\lambda1} + v_6^{P4,\lambda1}}{2}, v_7^{P4,\lambda2}, v_8^{P4,\lambda3}, v_9^{P4,\lambda4})$$

$$\overrightarrow{P5} = (\frac{v_6^{P5,\lambda1} + v_7^{P5,\lambda1}}{2}, v_8^{P5,\lambda2}, v_9^{P5,\lambda3}, ?)$$

$$\overrightarrow{P6} = (v_7^{P6,\lambda1}, v_8^{P6,\lambda2}, v_9^{P6,\lambda3}, ?)$$

Picture 10



Measured values

$v_{10}^{P4,\lambda4}$

$v_{10}^{P5,\lambda4}$

$v_{10}^{P6,\lambda3}$

Resulting vectors

$$\overrightarrow{P1} = (v_2^{P1,\lambda1}, v_3^{P1,\lambda2}, v_4^{P1,\lambda3}, v_5^{P1,\lambda4})$$

$$\overrightarrow{P2} = (v_3^{P2,\lambda1}, v_4^{P2,\lambda2}, v_5^{P2,\lambda3}, \frac{v_6^{P2,\lambda4} + v_7^{P2,\lambda4}}{2})$$

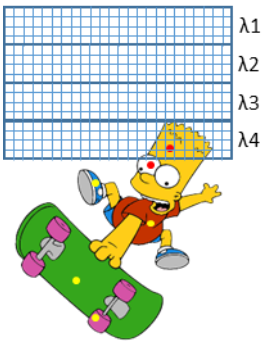
$$\overrightarrow{P3} = (v_4^{P3,\lambda1}, \frac{v_5^{P3,\lambda2} + v_6^{P3,\lambda2}}{2}, v_7^{P3,\lambda3}, v_8^{P3,\lambda4})$$

$$\overrightarrow{P4} = (\frac{v_5^{P4,\lambda1} + v_6^{P4,\lambda1}}{2}, v_7^{P4,\lambda2}, v_8^{P4,\lambda3}, \frac{v_9^{P4,\lambda4} + v_{10}^{P4,\lambda4}}{2})$$

$$\overrightarrow{P5} = (\frac{v_6^{P5,\lambda1} + v_7^{P5,\lambda1}}{2}, v_8^{P5,\lambda2}, v_9^{P5,\lambda3}, v_{10}^{P5,\lambda4})$$

$$\overrightarrow{P6} = (v_7^{P6,\lambda1}, v_8^{P6,\lambda2}, \frac{v_9^{P6,\lambda3} + v_{10}^{P6,\lambda3}}{2}, ?)$$

Picture 11



Measured values

$v_{11}^{P6,\lambda4}$

Resulting vectors

$$\overrightarrow{P1} = (v_2^{P1,\lambda1}, v_3^{P1,\lambda2}, v_4^{P1,\lambda3}, v_5^{P1,\lambda4})$$

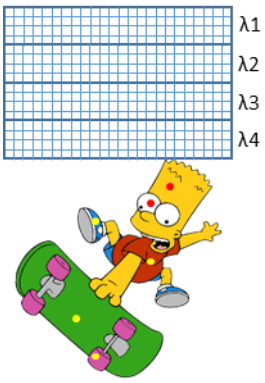
$$\overrightarrow{P2} = (v_3^{P2,\lambda1}, v_4^{P2,\lambda2}, v_5^{P2,\lambda3}, \frac{v_6^{P2,\lambda4} + v_7^{P2,\lambda4}}{2})$$

$$\overrightarrow{P3} = (v_4^{P3,\lambda1}, \frac{v_5^{P3,\lambda2} + v_6^{P3,\lambda2}}{2}, v_7^{P3,\lambda3}, v_8^{P3,\lambda4})$$

$$\overrightarrow{P4} = (\frac{v_5^{P4,\lambda1} + v_6^{P4,\lambda1}}{2}, v_7^{P4,\lambda2}, v_8^{P4,\lambda3}, \frac{v_9^{P4,\lambda4} + v_{10}^{P4,\lambda4}}{2})$$

$$\overrightarrow{P5} = (\frac{v_6^{P5,\lambda1} + v_7^{P5,\lambda1}}{2}, v_8^{P5,\lambda2}, v_9^{P5,\lambda3}, v_{10}^{P5,\lambda4})$$

$$\overrightarrow{P6} = (v_7^{P6,\lambda1}, v_8^{P6,\lambda2}, \frac{v_9^{P6,\lambda3} + v_{10}^{P6,\lambda3}}{2}, v_{11}^{P6,\lambda4})$$

Picture 12	Measured values	Resulting vectors
	-	$\overrightarrow{P1} = (v_2^{P1,\lambda1}, v_3^{P1,\lambda2}, v_4^{P1,\lambda3}, v_5^{P1,\lambda4})$ $\overrightarrow{P2} = (v_3^{P2,\lambda1}, v_4^{P2,\lambda2}, v_5^{P2,\lambda3}, \frac{v_6^{P2,\lambda4} + v_7^{P2,\lambda4}}{2})$ $\overrightarrow{P3} = (v_4^{P3,\lambda1}, \frac{v_5^{P3,\lambda2} + v_6^{P3,\lambda2}}{2}, v_7^{P3,\lambda3}, v_8^{P3,\lambda4})$ $\overrightarrow{P4} = (\frac{v_5^{P4,\lambda1} + v_6^{P4,\lambda1}}{2}, v_7^{P4,\lambda2}, v_8^{P4,\lambda3}, \frac{v_9^{P4,\lambda4} + v_{10}^{P4,\lambda4}}{2})$ $\overrightarrow{P5} = (\frac{v_6^{P5,\lambda1} + v_7^{P5,\lambda1}}{2}, v_8^{P5,\lambda2}, v_9^{P5,\lambda3}, v_{10}^{P5,\lambda4})$ $\overrightarrow{P6} = (v_7^{P6,\lambda1}, v_8^{P6,\lambda2}, \frac{v_9^{P6,\lambda3} + v_{10}^{P6,\lambda3}}{2}, v_{11}^{P6,\lambda4})$

The result of this scanning process are complete spectra for all object points considered.

Stitching the individual bands of consecutive wedge frames together will only produce a good result if:

The object moves under the sensor perpendicular to the orientation of the wedge bands. Any misalignment can result in incorrectly stitched images, both spatially and spectrally.

The frame rate at which the camera captures images is in sync with the speed at which the object is moved under the camera.

8.2.2. Maximum line rate calculation

The highest speed at which the camera can be moved without losing information is the height of a spectral band (in pixels) from image to image.

This value multiplied by the maximum frame rate gives the highest number of lines that can be read per second.

Height of a band:

- LS100: 8 pixels
- LS150: 5 pixels

Maximum frame rate:

- Standard xiQ-USB3.0 camera series: 170 fps @ 8 bit
- xiX PCIe camera series: 322 fps @ 10 bit

If the start-up and shutdown phases are not considered, the maximum line rate that can be achieved is:

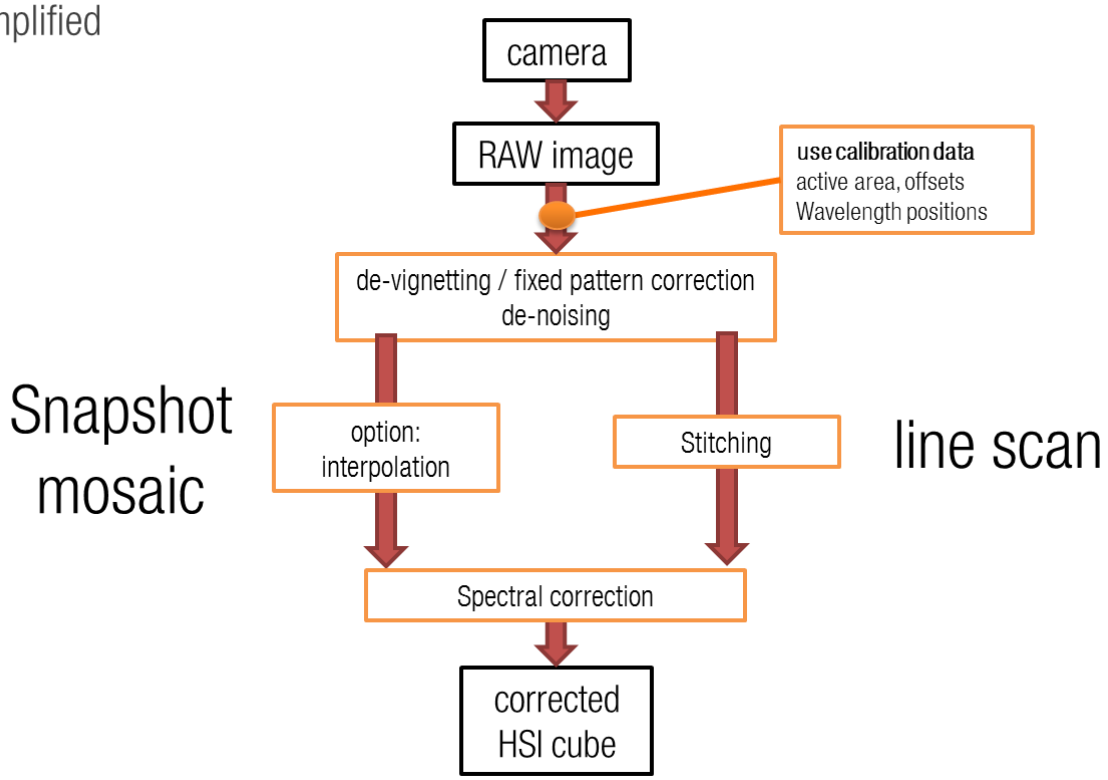
Height of a band (in pixels) * Frame rate [lines / s]

lines / s	xiQ	xiX
LS100	1 360	2 576
LS150	850	1 610

table 8-1, line scan sensors, max. line rate

9. Hyperspectral data correction

simplified



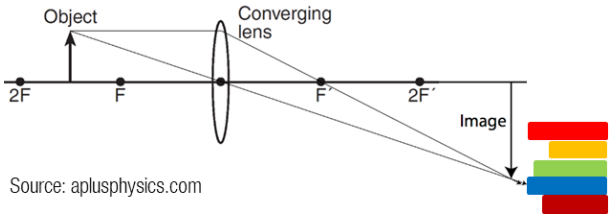
9.1. Data – spectral correction /snapshot mosaic sensor

These sensors have significantly different degrees of cross talks between neighboring pixels dependent on the angle of the light rays to perpendicular to the sensor surface.

The response curves are determined during sensor production and stored in the sensor-specific calibration file. The calibration data are delivered together with the cameras. The response curves, which also show the crosstalk between adjacent pixels, are determined with collimated light that falls vertically on the sensor surface.

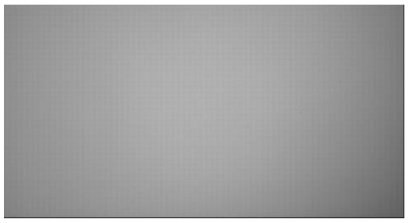
Cross talks (depending on the setup [camera, lens, aperture, distance to object and illumination]) can lead to changes of the spectral signature outside the sensor center:

Standard lens:



at high angles:
peak shift
band-specific vignetting

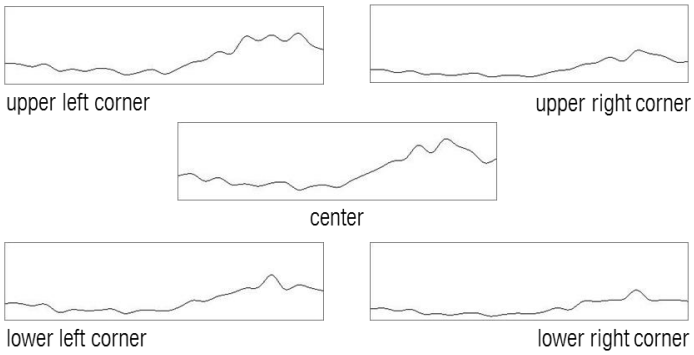
When using standard VIS-NIR lenses, a significant “vignetting” may occur:



f=2.8 (recommended aperture from imec)

Snapshot mosaic 5X5-NIR, 675-975nm,
Edmund Optics 35mm VIS-NIR lens
Halogen lighting

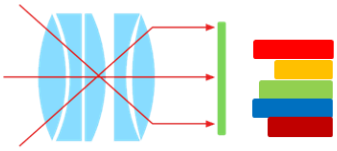
The “vignetting” has also an impact on the spectral curves:



It is recommended to a implement a white image / fixed pattern image correction for each band

This effect can be reduced using (sensor side) telecentric lenses:

Telecentric lenses:



Source: ivent.de

No peak shift and
band-specific vignetting

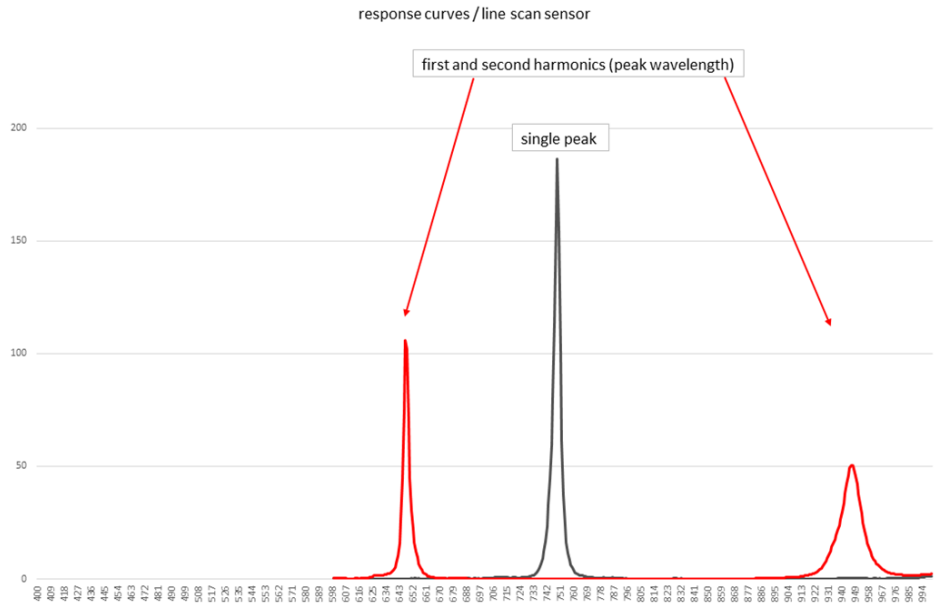
The reference values are measured by capturing images of a diffuse target, which reflects light of all wavelengths almost identically. (A diffuse reflectance target is part of XIMEA’s xiSpec2 starter kits).

This correction step is practically a band-specific “flat-field correction” in relation to the center of the sensor.

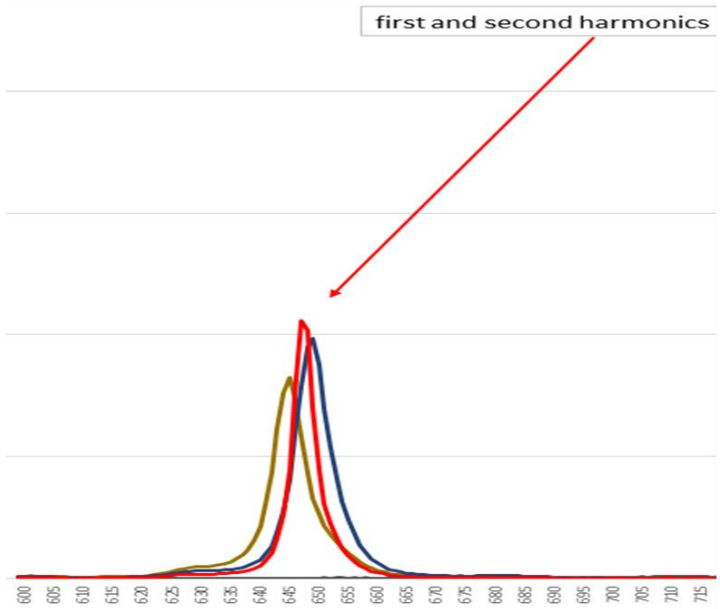
The cross talks can be corrected by multiplying the raw sensor spectra with a correction matrix.

9.2. Data – spectral correction / line scan mosaic

Some response curves have two peak wavelength (cannot be eliminated with long or short pass filters).



Some response curves have two peak wavelength (cannot be eliminated with long or short pass filters).



The position of the second harmonic (peak wavelength) is not the peak wavelength of another band.
This effect can be corrected by a correction matrix.

9.3. Requirements for the procedures described here

All following explanations / calculations assume that:

- the sensor is operated in a range of a linear ratio of exposure time to gray value in the RAW image.
- All measurements are performed when the sensor is already heated to operating temperature.

9.4. Band specific flat field correction

To correct the effect of brightness distribution of the lighting, the influence of the lens (vignetting, aperture) and especially the spectral shifts to the corners due to increasing crosstalk, a band-specific "flat field" correction $f^{b}_{x_P,y_P}$ can be applied.

Parameter / values:

$o_{X,S}$	Sensor filter area offset x
$o_{Y,S}$	Sensor filter area offset y
w_S	Width of the filter area
h_S	Height of the filter area
n_S	Pattern size (height and width) of the Fabry-Perot filter pattern
$W_S * H_S$	Spatial resolution (width x height - in snapshot mosaic pattern)
$2m+1$	Size of the center for reference calculation (usually: 10 <= m <= 20)
x_M	Center of the spatial resolution, x
y_M	Center of the spatial resolution, y
$V^{b}_{x_P,y_P}$	Raw value of band b (0-based index) of the snapshot-mosaic pattern position
	x_P, y_P with $0 <= x_P < W_S$ and $0 <= y_P < H_S$
$x^{b}_{x_P,y_P}$	Sensor x-position of $V^{b}_{x_P,y_P}$ (0-based index)
$y^{b}_{x_P,y_P}$	Sensor y-position of $V^{b}_{x_P,y_P}$ (0-based index)
V^{b}_{ref}	Reference value for band b
$f^{b}_{x_P,y_P}$	Correction value for pixel / band RAW value $V^{b}_{x_P,y_P}$

With:

$$x_M = \left[\frac{W_S}{2}\right]$$

$$y_M = \left[\frac{H_S}{2}\right]$$

$$x^{b}_{x_P,y_P} = o_{X,S} + x_P * n_S + b - \left[\frac{b}{n_S}\right] * n_S$$

$$y^{b}_{x_P,y_P} = o_{Y,S} + y_P * n_S + \left[\frac{b}{n_S}\right]$$

$$V^{b}_{ref} = \frac{\sum_{x=x_M-m}^{x_M+m} \sum_{y=y_M-m}^{y_M+m} V^{b}_{x,y}}{(2m+1)^2}$$

$$f^{b}_{x_P,y_P} = \frac{V^{b}_{ref}}{V^{b}_{x_P,y_P}}$$

$[.]$ Gauß bracket notation (round down to integer)

9.5. Reflectance calculation

The values read from the sensor is the captured radiance of an object.

The quantum efficiency (QE) of the sensor (sensor plus interference filter) depends on the wavelength. Radiance values therefore do not represent the light absorption or reflectance curves of materials used in literature.

In order to eliminate the influence of the entire setup (QE, transmission values of the optical path and the spatial and spectral variance of the illumination), the so-called reflectance can be calculated.

Basically, the object radiance is divided by a reference radiance. The reference values are measured by capturing images of a diffuse target, which reflects light of all wavelengths almost identically. These reflectance targets are often matte surfaces made of barium sulphate.

Due to different brightness levels of the object and the reference target, the exposure (integration) times may be different as well.

N	Number of spectral bands
$\vec{R} = (r_1, ..., r_N)$	Reflectance spectrum (vector), N -dimensional
$\vec{S}_O = (o_1, ..., o_N)$	Radiance spectrum (vector) of the object, N -dimensional
$\vec{S}_{Ref} = (\rho_1, ..., \rho_N)$	Radiance spectrum (vector) of the white reference target, N -dimensional
t_O	Exposure time of the object spectrum
t_{Ref}	Exposure time of the reference spectrum

$\vec{R} = (r_1, ..., r_N)$ is calculated as:

$$r_i = \frac{o_i}{\rho_i} \frac{t_{Ref}}{t_O}$$

The black value of the xiSpec2 cameras is usually set in such a way that it is not necessary to consider black images (when the camera is closed) for the reflectance calculation.

If dark images are to be used in the calculation after all, this can be achieved by using two dark images:

$\vec{D}_O = (d_1^O, ..., d_N^O)$	Dark image radiance spectrum, exposure time of the object spectrum
$\vec{D}_{Ref} = (d_1^{Ref}, ..., d_N^{Ref})$	Dark image radiance spectrum, exposure time of the reference spectrum

$\vec{R} = (r_1, ..., r_N)$ is then calculated as:

$$r_i = \frac{o_i - d_i^O}{\rho_i - d_i^{Ref}} \frac{t_{Ref}}{t_O}$$

For the calculation of the reflectance, the band specific flat-field correction described above can be omitted.

9.6. Spectral correction / Correction matrix

The application of this spectral correction requires a previously performed reflectance calculation!

Please note that the number of the generated (so called virtual) bands in the spectrum may vary depending on the sensor and filter set used. As mentioned before: In some cases, the peak wavelengths are very close, the response curves overlap almost completely.

In such cases, one effective spectral band will be generated to allow quantitative analyses. The number of spectral bands contained in the corrected spectrum can therefore be lower than the number of pixels in the sensor pattern.

N_S Number of bands on the sensor surface

N_C Number of the (virtual) bands in the corrected spectrum

$M^{N_C \times N_S}$ Correction matrix $M^{N_C \times N_S}$ with N_C rows and N_S columns

$\overrightarrow{S_S}$ Uncorrected (reflectance / sensor) spectrum (vector) with N_S elements

$\overrightarrow{S_C}$ Corrected spectrum (vector) with N_C elements

$\overrightarrow{S_C}$ is calculated as:

$$\overrightarrow{S_C} = M^{N_C \times N_S} \times \overrightarrow{S_S}$$

Please note:

- The elements $\lambda_{C,n}$ in $\overrightarrow{S_C} = \left(\lambda_{S,1}, \ldots, \lambda_{S,N_C}\right)$ are sorted by wavelength.
- The elements $\lambda_{S,n}$ in $\overrightarrow{S_S} = \left(\lambda_{S,1}, \ldots, \lambda_{S,N_S}\right)$ are sorted by their sensor pattern position (index+1) (!) to use the regular correction matrix $M^{N_C \times N_S}$ (stored in the calibration file).

If $\overrightarrow{S_S}$ is sorted by wavelength either the elements in $\overrightarrow{S_S}$ or the columns in $M^{N_C \times N_S}$ must be re-sorted!

9.6.1. Example correction data / virtual bands (part of the sensor calibrations files):

N_C (virtual) bands of the corrected spectrum are defined in the section <correction_matrix> in the sensor calibration file:

```
<correction_matrices>
  <correction_matrix ….>
  . . . . .
        <virtual_bands>
              <virtual_band … >
                    <wavelength_nm> λ_v,1 </wavelength_nm>
                    <fwhm_nm> ϖ_1 </fwhm_nm>
                    <coefficients nr_elements=" N_S " values=" f_1,1  f_1,2  f_1,3  ..  f_1,N_S " />
              </virtual_band>
              <virtual_band … >
                    <wavelength_nm> λ_v,2 </wavelength_nm>
                    <fwhm_nm> ϖ_2 </fwhm_nm>
                    <coefficients nr_elements=" N_S " values=" f_2,1  f_2,2  f_2,3  ..  f_2,N_S " />
              </virtual_band>
              <virtual_band … >
                    <wavelength_nm> λ_v,3 </wavelength_nm>
                    <fwhm_nm> ϖ_3 </fwhm_nm>
                    <coefficients nr_elements=" N_S " values=" f_3,1  f_3,2  f_3,3  ..  f_3,N_S " />
              </virtual_band>
              . . . . .
              . . . . .
              <virtual_band … >
                    <wavelength_nm> λ_v,N_C </wavelength_nm>
                    <fwhm_nm> ϖ_N_C </fwhm_nm>
                    <coefficients nr_elements=" N_S " values=" f_N_C,1  f_N_C,2  f_N_C,3  ..  f_N_C,N_S " />
              </virtual_band>
        </virtual_bands>
  </correction_matrix>
</correction_matrices>
```

The correction matrix is:

$$M^{N_C \times N_S} = \begin{pmatrix} f_{1,1} & f_{1,2} & f_{1,3} & .. & .. & f_{1,N_S} \\ f_{2,1} & f_{2,2} & f_{2,3} & .. & .. & f_{2,N_S} \\ f_{3,1} & f_{3,2} & f_{3,3} & .. & .. & f_{3,N_S} \\ .. & .. & .. & .. & .. & .. \\ f_{N_C,1} & f_{N_C,2} & f_{N_C,3} & .. & .. & f_{N_C,N_S} \end{pmatrix}$$

Uncorrected (sensor) spectrum:

The elements $\lambda_{S,n}$ in $\overrightarrow{S_S}$ are sorted by their sensor pattern position:

1	2	..	..
..	..	..	..
..	..	..	..
..	..	..	N_S

$$\overrightarrow{S_S} = \left(\lambda_{S,1},...,\lambda_{S,N_S}\right)$$

Corrected spectrum

The elements $\lambda_{C,n}$ in $\overrightarrow{S_C}$ are sorted by wavelength. These wavelengths are the peaks of the virtual bands (see the example correction matrix / virtual bands above).

$$\overrightarrow{S_C} = M^{N_C \times N_S} \times \left(\lambda_{S,1},...,\lambda_{S,N_C}\right) = \left(\lambda_{v,1},...,\lambda_{v,N_C}\right) \quad \text{[virtual band peaks]}$$

10. Development:

A software solution must be used / developed that is able to acquire the RAW data from the camera and read and interpret the calibration date. An API/SDK to acquire the RAW data and control the camera is available for free. Please note that additional software components or own image processing is required to generate the so-called hyperspectral cubes.

10.1. API / SDK / drivers:

Our API / drivers etc. are available for download for free:

Most recent beta version: https://www.ximea.com/support/documents/14

Stable version (and LINUX / MacOS files): https://www.ximea.com/support/documents/4

Link to our API: https://www.ximea.com/support/wiki/apis/APIs

C/C++ manual: https://www.ximea.com/support/wiki/apis/XiAPI_Manual

C# manual: https://www.ximea.com/support/wiki/apis/XiAPINET_Manual

Part of the driver/API is the standard viewer xiCamTool.

After installation of the API you'll find some samples (with complete source code) at:

C:\XIMEA\Examples (Windows):

Example\xiSample	C: how to start
Example\Sources\xiAPI	C#: how to start

Please do not use MONO modes. Only RAW8 or RAW16 will deliver the original sensor data. MONO8 / MONO16 is a post-processed format. Please note:

https://www.ximea.com/support/projects/apis/wiki/XiApi_Manual#XI_PRM_IMAGE_DATA_FORMAT-or-imgdataformat

You can set the output bit depth using:

https://www.ximea.com/support/projects/apis/wiki/XiApi_Manual#XI_PRM_SENSOR_DATA_BIT_DEPTH-or-sensor_bit_depth

https://www.ximea.com/support/projects/apis/wiki/XiApi_Manual#XI_PRM_OUTPUT_DATA_BIT_DEPTH-or-output_bit_depth

https://www.ximea.com/support/projects/apis/wiki/XiApi_Manual#XI_PRM_IMAGE_DATA_BIT_DEPTH-or-image_data_bit_depth

It is recommended to use the data packing to increase the transfer speed in case of a bit depth of 10 or 12 bits.

If you need to change the exposure time during your operation, please use the parameter:

https://www.ximea.com/support/projects/apis/wiki/XiApi_Manual#XI_PRM_IMAGE_DATA_BIT_DEPTH-or-image_data_bit_depth

10.2. Access to the sensor calibration files

IMEC measures every multi-/hyperspectral sensor and xiSpec2 camera individually. The resulting calibration files will be delivered together with the sensor / camera.

XIMEA stores the calibration files in the file system of the camera. The files can be read using API functions. Additionally, the calibration files are delivered on a USB-stick. The calibration files in the camera file system are stored zipped.

10.2.1. File name schematics

The filename structure is:

CMV2K-<Type>-<W-Range>-<SENS-SerNr>.xml

Type: sensor type:

- SSM4x4 Snapshot mosaic, pattern size 4x4, 16 interference filters
- SSM5x5 Snapshot mosaic, pattern size 5x5, 25 interference filters
- (LS100 Line scan, 100 bands NIR – discontinued)
- LS150 Line scan, 150 bands VISNIR
-

W-Range: Active wavelength range:

- 470_620 SM4x4 VIS2 camera
- 460_600 SM4x4 VIS3 camera
- 595_860 SM4x4 RN2 camera
- 665_975 SM5x5 NIR2 camera
- 600_1000 LS100 NIR2
- 470_900 LS150 VN2

SENS-SerNr: Sensor serial number

- 4 numbers, separated by dots, e.g. 4.2.16.0

Examples for valid filenames:

- CMV2K-SSM5x5-665_975-1.2.3.4.xml
- CMV2K-SSM4x4-460_600-1.2.3.4.xml
- CMV2K-LS150-470_900-1.2.3.4.xml

10.2.2. File storing in the file system

There is exactly one calibration file for one xiSpec2 camera. The calibration file is stored zipped in the file system of the camera. To simplify access, the same file name is always used:

hyperspectral_cal_data

For some first cameras of the xiSpec2 series, other file names were used for the calibration files: the file names as described in chapter *10.2.1 File name schematics* with the file extension *.zip, example:

CMV2K-SSM4x4-470_620-9.2.4.11.zip

To remain compatible with previous implementations and to support the possibility of future extensions of the concept, additional file with mapping information is stored:

sens_calib.dat

Directory of the file system, Example:

```
Directory of 'mmf1/*'
13-Jan-2021 12:11:58         63016 hyperspectral_cal_data
13-Jan-2021 12:11:58           169 sens_calib.dat
          2 files
      63185 bytes
```

10.2.3. Mapping info, sens_calib.dat

The file sens_calib.dat is stored in XML-format. The XSD structure is listed in Appendix *13.2.1 Calibration file mapping XML Schema V1.0.1*.

The content of the file sens_calib.dat is a translation table IMEC calibration file name <-> file name in cameras file system:

<file_name> is the name of the calibration file

<file_link> is the name of the file in the file system

Example:

```
<calibrations>
<calibration>
     <file_name>CMV2K-SSM4x4-470_620-9.2.4.11.xml</file_name>
     <file_link> hyperspectral_cal_data</file_link>
</calibration>
</calibrations>
```

10.2.4. API implementation

10.2.4.1. API function to get the sensor serial number which is used to build the file names:

C / C++

```
char  sensor_sn [100] = "";
xiGetParamString(xiH, XI_PRM_DEVICE_SENS_SN , sensor_sn, 99);
```

.NET

```
String xi_sensor_sn;
xiH.GetParam(PRM.DEVICE_SENS_SN, out xi_sensor_sn);
```

10.2.4.2. API function to read files from the file system

C / C++

Set the file name to be used

```
char  xi_filename[MAX_PATH]="sens_calib.dat";  // e.g.
xiSetParamString(xiH, XI_PRM_FFS_FILE_NAME, xi_filename, sizeof (xi_filename));
```

Read a file

```
#define BUFFER_LENGTH   1024000
char  *buffer_InOut;
int   i_filelength = 0;
buffer_InOut = (char*) calloc(BUFFER_LENGTH, 1);
xiGetParamString(xiH, XI_PRM_READ_FILE_FFS, buffer_InOut, BUFFER_LENGTH);
```

.NET

Set the file name to be used

```
String internalFileName_used;
internalFileName_used = "sens_calib.dat"; // e.g.
xiH.SetParam(PRM.FFS_FILE_NAME, internalFileName_used);
```

read a text file:

```
String buffer;
buffer = "";
xiH.GetParam(PRM.READ_FILE_FFS, out buffer);
fileLength = buffer.Length;
```

read a binary file (zip-container)

```
int file_size = xiH.GetParamInt(PRM.FFS_FILE_SIZE);
byte[] bufferZipped = new byte[file_size + 1];
xiH.GetParam(PRM.READ_FILE_FFS, out bufferZipped, out file_size);
```

10.2.4.3. API functions to read the directory from the file system

Example code to get the info about the files in the file system:

.NET

```
class DirectoryEntry
{
    public string name;
    public int size;
}
List<DirectoryEntry> directory = new List<DirectoryEntry>();

int free_ffs_size, used_ffs_size, maxID_directory;
myCam.GetParam(PRM.FREE_FFS_SIZE, out free_ffs_size);
myCam.GetParam(PRM.USED_FFS_SIZE, out used_ffs_size);

int maxID_directory = myCam.GetParam(PRM.FFS_FILE_ID_MAX);

for (int ii=0; ii<=maxID_directory; ii++)
{
      DirectoryEntry directoryEntry = new DirectoryEntry();

      myCam.SetParam(PRM.FFS_FILE_ID, ii);
      string buffer = myCam.GetParamString(PRM.FFS_FILE_NAME)
      int fileLength = myCam.GetParamInt(PRM.FFS_FILE_SIZE);

      directoryEntry.name = buffer;
      directoryEntry.size = fileLength;
directory.Add(directoryEntry);
}
```

10.2.4.4. Example .NET project to read the calibration data

The XML Schemas can be used to generate directly usable c#-classes easily (the namespace “xiHsi” is used for instance only):

xsd sens_calib.xsd /c /f /n:xiHsi

xsd HSI_calibration_V2.0.1.xsd /c /f /n:xiHsi

Prepare a new Visual Studio project:

Framework .Net Framework 4.7.2

Activate framework reference:

Assemblys\Framework\System.IO.Compression

Add a new reference to get access to XIMEAs camera API:

Browse\xiAPI.NET64.dll

path: C:\XIMEA\API\xiAPI.NET.Framework.4.7.2\xiApi.NETX64.xml

Change a program property:

Build: Platform: x64

```
namespace xiHsi {
    using System.Xml.Serialization;

// c#-classes generated from the XML-Schemas:
    public partial class Calibrations { … }
    public partial class Sensor_Calibration { … }
}

…
using System.IO;
using System.IO.Compression;
using System.Xml.Serialization;
using xiApi.NET;
….

Sensor_Calibration sensor_calibration;

xiCam xiH = new xiCam(); ;
xiH.OpenDevice(0);            // first camera in enumeration

xiH.SetParam(PRM.FFS_FILE_NAME, "sens_calib.dat");
xiH.GetParam(PRM.READ_FILE_FFS, out string sens_calib_buffer);

XmlSerializer serializer_Cal = new XmlSerializer(typeof(Calibrations));
Calibrations calibrations
= (Calibrations)serializer_Cal.Deserialize(new StringReader(sens_calib_buffer));

xiH.SetParam(PRM.FFS_FILE_NAME, calibrations.calibration[0].file_link);

int file_size = xiH.GetParamInt(PRM.FFS_FILE_SIZE);
byte[] bufferZipped = new byte[file_size + 1];
xiH.GetParam(PRM.READ_FILE_FFS, out bufferZipped, out file_size);

ZipArchive archive = new ZipArchive(new MemoryStream(bufferZipped));
ZipArchiveEntry entry = archive.Entries[0];
String calibration_buffer = (new StreamReader(entry.Open())).ReadToEnd();

XmlSerializer serializer = new XmlSerializer(typeof(Sensor_Calibration));
sensor_calibration
= (Sensor_Calibration)serializer.Deserialize(new StringReader(calibration_buffer));

xiH.CloseDevice();
```

11. Measurement protocol

At the end of the manufacturing process, the sensor, the bandpass filter and the entire camera (together with the recommended lenses) are calibrated and checked against the targets and maximum tolerances as defined above in Filter layouts / camera models *4 Filter layouts / camera models*.

The result is cameras that can be used as spectral measurement instruments.

Each camera is accompanied by a measurement protocol showing the comparison of spectral curves of the camera with a high-resolution point spectrometer. Standardized color measurement cards are used for the measurement:

File name conventions for the different models:

Ximea_<Layout>_< W-Range >_<NrBands>_<imec-desr>_<CameraSerNr>_<SensorSerNr>.pdf

<Layout>:	Layout of the Interference filter
snapshot	snapshot mosaic sensor / camera
linescan	line scan sensor / camera
<W-Range>:	Wavelength range
VIS	visible light
RedNIR	Red and NIR
NIR	NIR
VNIR	Visible light to NIR
<NrBands>:	Number of spectral bands (after spectral correction)
<imec-descr>	imec internal sensor description
<CameraSerNr>:	Camera Serial number
<SensorSerNr>:	Sensor serial number, 4 numbers, separated by dots, e.g. 4.2.16.0

Example (SM4X4-RN2 camera model):

Ximea camera 39070261 with imec snapshot RedNIR sensor

Camera Info

Spectral Quality

Camera Info

✔ General Camera Info

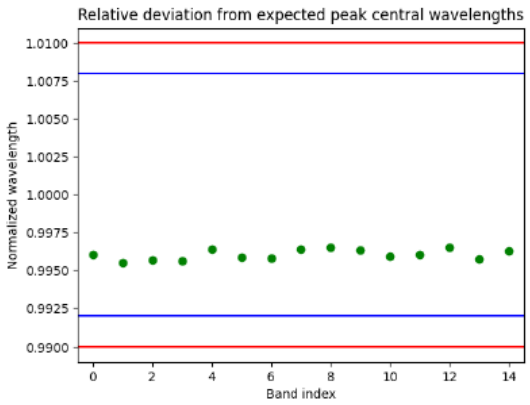
Imec part number	360-0002-01
Camera type	snapshot RedNIR
Number of samples in detection range	15
Detection range	600-860
Band pass filter type	internal filter
Bit depth	10 bit
Recommended lens	exit pupil telecentric with f-number ≥ 2.8

✔ Camera-Specific Info

Camera ID	39070261
Sensor ID	13.10.6.8
Calibration filename	CMV2K-SSM4x4-595_860-13.10.6.8.xml

Peak central wavelength info

The peak central wavelengths of the sensor under collimated light are compared to the expected peak central wavelengths (see table below the plot). The measured peak central wavelengths are checked to be all within +/-1.0% (red lines) and on average within +/-0.8% (blue lines) from the expected wavelengths.



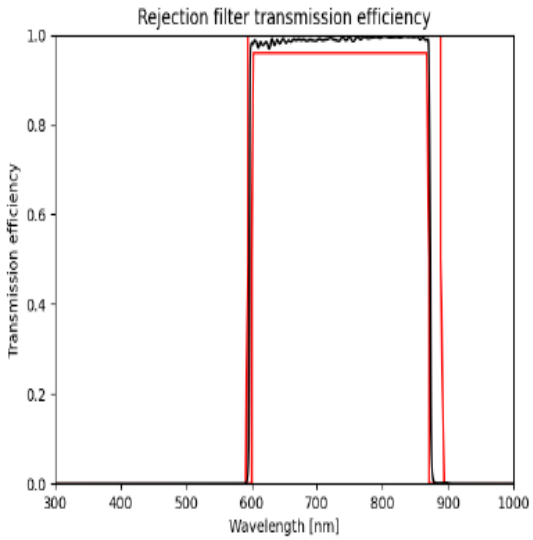
Expected peak central wavelengths under collimated light and % deviation of the measured w.r.t. the expected peak central wavelengths:

Band index	Expected wavelengths [nm]	Measured wavelengths [nm]	% shift	Band index	Expected wavelengths [nm]	Measured wavelengths [nm]	% shift
0	609.0	606.6	-0.4	8.0	754.1	751.5	-0.3
1	625.6	622.8	-0.4	9.0	770.1	767.3	-0.4
2	648.0	645.2	-0.4	10.0	786.2	783.0	-0.4
3	666.3	663.4	-0.4	11.0	802.4	799.2	-0.4
4	683.9	681.4	-0.4	12.0	818.3	815.4	-0.3
5	700.8	697.9	-0.4	13.0	833.1	829.6	-0.4
6	718.9	715.9	-0.4	14.0	849.4	846.2	-0.4
7	736.6	733.9	-0.4				

The average deviation in peak central wavelength is -0.4%.

Band pass filter info

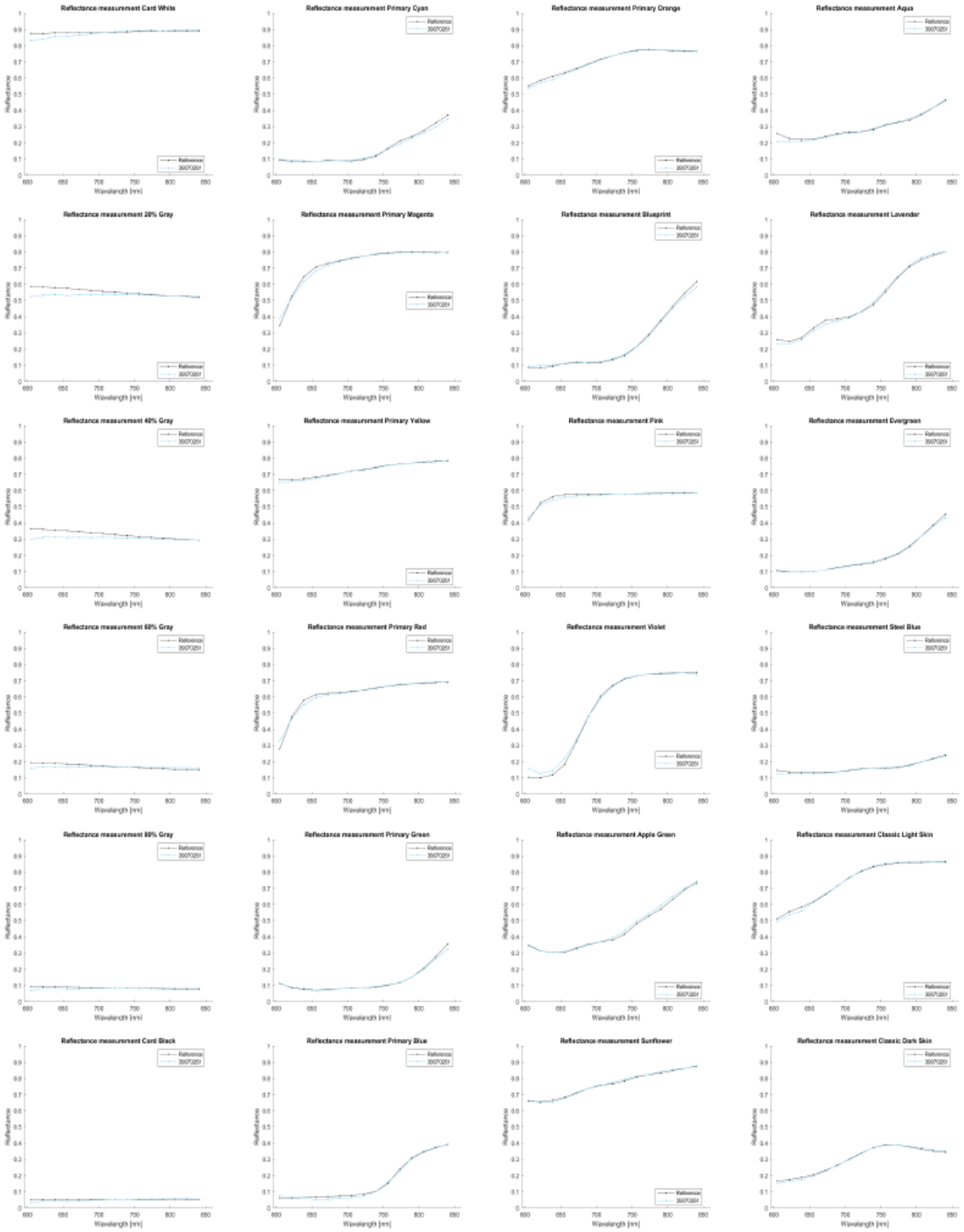
The measured transmission efficiency of the camera's rejection filter is plotted below (black line). The red lines indicate the accepted shift in cut-on/cut-off wavelength and the minimally accepted transmission efficiency.



Spectral Quality

A set of calibrated reference reflectance targets are measured under broadband LED light with the current camera and the spectra are compared to the reference spectra.

✔ **Spectral measurements** V



✔ Accuracy reflectance spectra V

Table with the similarity index computed between the reconstructed and the reference reflectance spectrum for each reference tile:

Card White	0.068411	Primary Cyan	0.045145	Primary Orange	0.031226	Aqua	0.055244
20% Gray	0.11347	Primary Magenta	0.063329	Blueprint	0.049432	Lavender	0.056305
40% Gray	0.11893	Primary Yellow	0.028076	Pink	0.038242	Evergreen	0.027539
60% Gray	0.060314	Primary Red	0.062512	Violet	0.078236	Steel Blue	0.029308
80% Gray	0.033166	Primary Green	0.037096	Apple Green	0.047876	Classic Light Skin	0.038726
Card Black	0.030858	Primary Blue	0.039349	Sunflower	0.024478	Classic Dark Skin	0.026489

This report was generated on 2021-01-20 with the imec HSI Systems Pro software v1.1.0.0.

12. Software and support (imec)

A software solution must be used / developed that is able to capture the RAW data from the camera and read and interpret the calibration data. XIMEA's API/SDK to acquire the RAW data and control the camera is available for free.

As a result of the intensified collaboration between Imec and XIMEA, buyers of xiSpec2 (and newer models) Snapshot Mosaic cameras have access to Imec's HSI-Mosaic software:

- HSI Mosaic: GUI-based acquisition software.
- HSI Camera API: connect, configure and capture raw data from a camera
- HSI Mosaic API: Configure and use a processing pipeline for Imec Mosaic data

Registered xiSpec2 camera customers have direct access to Imec's first-line support for hyperspectral imaging questions, Imec's manuals and sample data. Imec provides tailored, expert-level technical support to help customers make the right technical decision and allow them to focus on their application.

The most recent version and the software manual can be downloaded from imec's support pages after registration.

Only a brief overview is given here:

12.1. Imec HSI Mosaic GUI software for snapshot systems

imec provides acquisition software "HSI Mosaic" with an intuitive user interface to enable users to easily acquire spectral data of the highest possible quality with imec snapshot camera systems. This entails the full chain of data acquisition from a camera, data referencing, demosaicing and all corrections required to create hyperspectral data cubes.

The workflow is split in two main steps:

1. Acquire: configure the camera and acquisition parameters to capture data
2. Analyze: configure the data processing parameters, interact with the data and export

To maximize ease of use. this workflow is embedded in features to automatically detect and easily connect to available cameras as well as to explore machine learning techniques for data analysis.

The "HSI Mosaic" software runs on PCs with windows 10 x64.

12.1.1. Acquire

The Acquire step enables setting camera parameters and acquisition of raw data from the camera. A live view is provided with different rendering options to enable correctly choosing viewport, focus and exposure time.

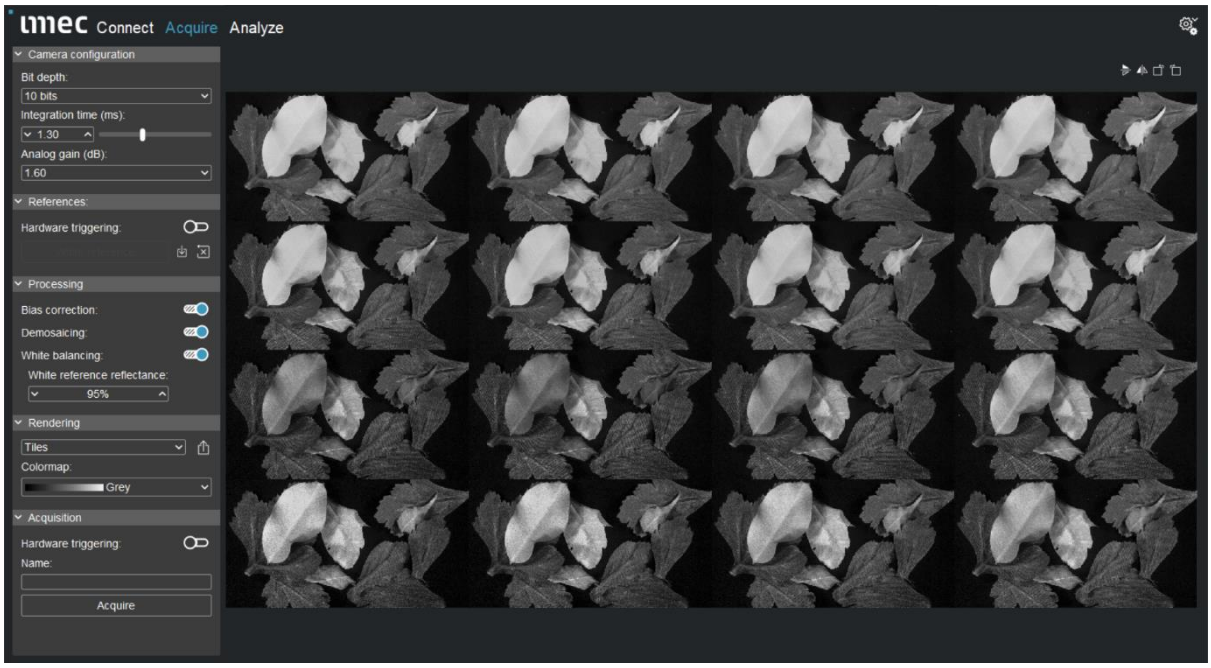


Figure 12-1, HSI Mosaic in the Acquire mode.

Acquisition parameters can be set in the left panel while a live view on the raw data is provided in the center. The data shown was acquired with a snapshot RedNIR camera.

Functionality:

- Detect and enumerate connected devices
- Link a device to calibration file
- Connect a specific device
- Set bit depth and integration time
- Live view on the raw camera data with multiple renderings
- Set parameters of optical system for spectral lens corrections
- Acquisition of reference data (bright field, dark field)
- HW / SW triggering

12.1.2. Analyze

In the Analyze step raw data is processed into fully processed irradiance or reflectance data cubes. The pipeline can process data captured in the Acquire step, or live data directly streaming from the camera or data acquired during a previous session. The fully processed spectral data is visualized as rendered images and in a spectrograph.

Functionality:

- Select the data source: workspace, live stream, offline
- Configure processing parameters:
  - Spatial median filter
  - Spatial median filter
  - Demosaicing output image resolution
  - Spectral balancing
- Tune image rendering: Color, false color, single band, tiled, NDVI
  - With gain, gamma and contrast sliders and rendering specific options
- Annotate and select regions with various shapes
- Visualize spectra in the spectrograph
- Simple imec SAM classifier and integration with third-party ML tools
- Extensive export options:
  - Processed cube data in ENVI or individual band images
  - RAW frame data with context information for offline (re-)processing
  - Individual spectra or averages of selections
  - Rendering, class image, plots
  - Selection / annotation mask, classifier centroids

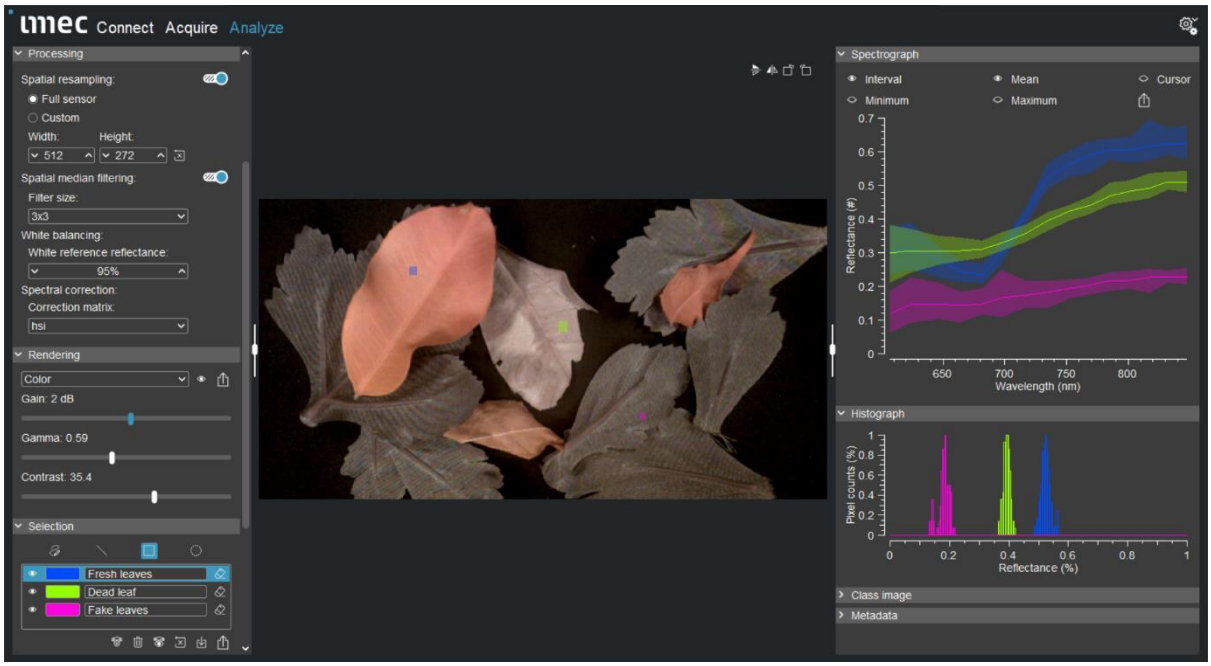


Figure 12-2, HSI Mosaic in the Analyze mode.

The raw data is processed and visualized as a broadband color image with spectra plotted for the selections made by the user. The processing pipeline parameters are configurable in the left panel. The data shown was acquired with a snapshot RedNIR camera.

12.2. Imec HSI API

imec API offering for snapshot systems

imec provides application programming interfaces (APIs) to enable the end-to-end hyperspectral imaging in custom software. This entails the full chain of data acquisition from a camera, data referencing, demosaicing and all corrections required to create hyperspectral data cubes.

The implementation is split over two APIs:

- Camera API to configure and control the camera for data acquisition
- HSI Mosaic API to process the imec mosaic data

Both APIs are available in C and Python and target a windows x64 environment. A compilation against a different platform can be provided on demand for e.g., embedded systems.

12.3. Camera API

The Camera API enables setting camera parameters and acquisition of raw data from the camera.

Functionality:

- Detect and enumerate connected devices
- Logging
- Connect / disconnect a specific device
- Get / set parameters
- Specify regions of interest
- Set high dynamic range (HDR) exposure times per pixel
- Data allocation
- Trigger
- Poll for data
- Save / load data to disk

Parameters:

- Exposure time
- Frame rate
- Triggering (fixed frame rate / external hardware trigger / software trigger)
- Sensor bit depth (8 / 10 bit)
- Temporal averaging
- Frame flipping (horizontal / vertical)
- Region of interest (unlimited along sensor height)
- HDR exposure times per pixel

12.4. HSI Mosaic API

The HSI Mosaic API provides access to the processing pipeline to transform raw data into fully processed radiance or reflectance data cubes. The input of the processing pipeline are the raw data acquired with the Camera API. The output are spectral data cubes.

An overview of the pipeline is illustrated below.

Functionality:

- Initialize the pipeline from the sensor calibration file
- Set the dark field image to remove data offset
- Full resolution demosaicing with band alignment
- Median filtering in the spatial domain
- Set the bright field image for non-uniformity correction (Note: requires the imec HSI evaluation setup)
- Set the lens parameters (focal length, exit pupil location) for angularity correction
- Provide a reference to white-balance or convert the radiance data into reflectance
- Save / load data to disk in standard ENVI file format

Parameters:

- Demosaicing output resolution
- Median filter size (disabled, 3x3, 5x5)
- Lens parameters
- Reference spectrum reflectance

12.5. Hints to install imec's HSI suite

(as of 2021-06-15)

Start the

HSI Suite Installer.exe

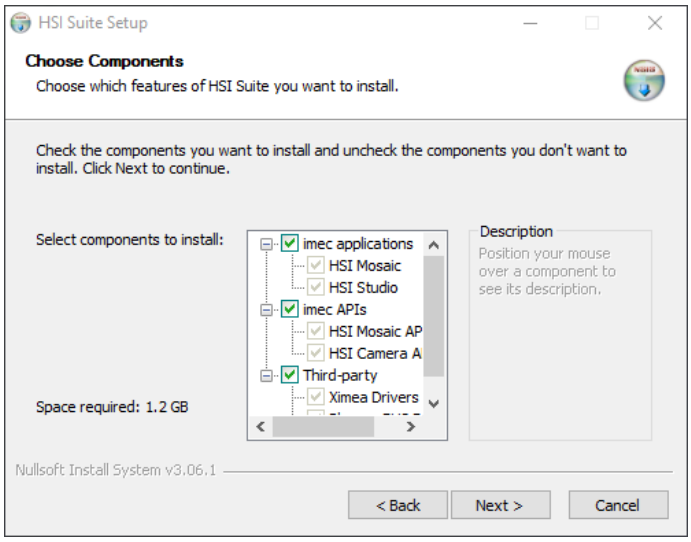
It may happen, hat Windows refuses to start the software:

Please click "more info" to continue:

And "run anyway" (Trotzdem ausführen):

Please accept the start of the application and the license agreement.

The next step “choose components” does not work as expected: None of the items can be disabled.

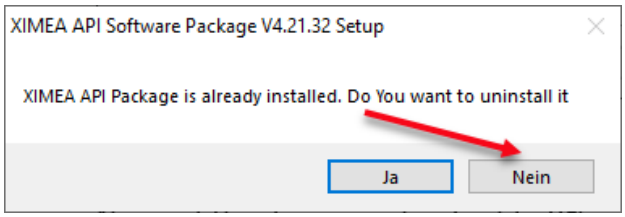


Imec applications:	needed
Imec APIs:	only needed for software development
Third-Party:	XIMEA Drivers are needed (please see below)

PLEASE NOTE:

The XIMEA drivers are needed for the camera communication.

The XIMEA drivers may already be installed on the computer, e.g. if other cameras from XIMEA are used. If a XIMEA API > V4.21.32 is installed, the installation of the API delivered with imec’s HSI suite should be skipped:



Please read the README.txt file in the folder “C:\imec\HSI Suite” and the manual

\imec-HSI-Suite\HSI Mosaic Software Manual.pdf

for more info.

13. Appendix

13.1. Copyright

13.2. XML Schema

According to W3C XML Schema Definition Language (XSD) the namespace prefix xs id bound to the standard namespace http://www.w3.org/2001/XMLSchema.

All XSD structures are described at https://www.w3.org/TR/xmlschema11-1/ and the datatypes used are described in http://www.w3.org/TR/xmlschema11-2/.

Further documents can be found at https://www.w3.org/TR/?title=xml%20schema.

The XML Schema XSD descriptions files are stored on the USB-stick which is part of the camera delivery.

13.2.1. Calibration file mapping XML Schema V1.0.1

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <!-- xiSpec2 camera series file mapping file format, XIMEA -->
    <!-- V1.0.1, 2021-04-12 -->

    <xs:element name="calibrations" type="Calibrations"/>

    <xs:complexType name="Calibrations">
        <xs:sequence>
            <xs:element name="calibration" type="Calibration" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="Calibration">
        <xs:sequence>
            <xs:element name="file_name" type="xs:string" />
            <xs:element name="file_link" type="xs:string" />
        </xs:sequence>
    </xs:complexType>

</xs:schema>
```

13.2.2. Calibration files XML Schema V2.0.1

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <!-- xiSpec2 camera series calibration file format, XIMEA -->
    <!-- V2.0.1, 2021-04-12 -->

    <!-- calibration file objects -->
    <xs:element name="sensor_calibration" type="Sensor_Calibration"/>

    <xs:complexType name="Sensor_Calibration">
        <xs:sequence>
            <xs:element name="sensor_info" type="Sensor_Info"/>
            <xs:element name="filter_info" type="Filter_Info"/>
            <xs:element name="system_info" type="System_Info"/>
        </xs:sequence>
        <xs:attribute name="version" type="xs:int" use="required" fixed="3" />
        <xs:attribute name="sensor_id" type="xs:string" use="required" />
        <xs:attribute name="created" type="xs:dateTime" use="required" />
        <xs:attribute name="modified" type="xs:dateTime" use="required" />
        <xs:attribute name="software" type="xs:string" use="required" />
        <xs:attribute name="software_version" type="xs:string" use="required" />
    </xs:complexType>


    <!-- Enumerations -->
    <xs:simpleType name="Filter_Layout_Enum">
        <xs:restriction base="xs:string">
```

```
            <xs:enumeration value="MOSAIC"/>
            <xs:enumeration value="TILED"/>
            <xs:enumeration value="WEDGE"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="Peak_Shape_Enum">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Fabry-Perot"/>
            <xs:enumeration value="Gaussian"/>
            <xs:enumeration value="Lorentzian"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="Optical_Component_Type_Enum">
        <xs:restriction base="xs:string">
            <xs:enumeration value="shortpass_filter"/>
            <xs:enumeration value="longpass_filter"/>
            <xs:enumeration value="bandpass_filter"/>
            <xs:enumeration value="notch_filter"/>
            <xs:enumeration value="linear_polarizer"/>
            <xs:enumeration value="circular_polarizer"/>
            <xs:enumeration value="source"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="Correction_Matrix_Algorithm_Enum">
        <xs:restriction base="xs:string">
            <xs:enumeration value="m0"/>
            <xs:enumeration value="m1"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="Correction_Matrix_Type_Enum">
        <xs:restriction base="xs:string">
            <xs:enumeration value="irradiance"/>
            <xs:enumeration value="reflectance"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="Sensor_Info_Type_Enum">
        <xs:restriction base="xs:string">
            <xs:enumeration value="CMV2K" />
            <xs:enumeration value="CDL640" />
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="Gain_Mode_Type_Enum">
        <xs:restriction base="xs:string">
            <xs:enumeration value="low" />
            <xs:enumeration value="medium" />
            <xs:enumeration value="high" />
        </xs:restriction>
    </xs:simpleType>


    <!-- double value argument lists -->
    <xs:simpleType name="doubleList">
        <xs:list itemType="xs:double"/>
    </xs:simpleType>

    <xs:complexType name="Value_List_Double">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="nr_elements" type="xs:int" use="required"/>
                <xs:attribute name="values" type="doubleList" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>



    <!-- main object sensor_info-->
    <xs:complexType name="Sensor_Info">
        <xs:sequence>
            <xs:element name="width_px" type="xs:int" />
            <xs:element name="height_px" type="xs:int" />
            <xs:element name="pixel_pitch_um" type="xs:double" />
            <xs:element name="bit_depth" type="xs:int" />
            <xs:element name="overall_gain" type="xs:double"  />
            <xs:element name="analog_gain" type="xs:double" minOccurs="0" />
            <xs:element name="digital_gain" type="xs:double" minOccurs="0" />
            <xs:element name="full_well_capacity_e" type="xs:int" minOccurs="0" />
            <xs:element name="gain_mode" type="Gain_Mode_Type_Enum" minOccurs="0" />
        </xs:sequence>
        <xs:attribute name="version" type="xs:int" use="required" fixed="2" />
        <xs:attribute name="sensor_type" type="Sensor_Info_Type_Enum" use="required" />
    </xs:complexType>
```

```
<!-- main object filter_info-->
<xs:complexType name="Filter_Info">
    <xs:sequence>
        <xs:element name="calibration_info" type="Calibration_Info"/>
        <xs:element name="filter_zones" type="Filter_Zones"/>
    </xs:sequence>
    <xs:attribute name="version" type="xs:int" use="required" fixed="1" />
</xs:complexType>

<xs:complexType name="Calibration_Info">
    <xs:sequence>
        <xs:element name="sample_points_nm" type="Value_List_Double"/>
    </xs:sequence>
    <xs:attribute name="version" type="xs:int" use="required" fixed="5" />
</xs:complexType>

<xs:complexType name="Filter_Zones">
    <xs:sequence>
        <xs:element name="filter_zone" type="Filter_Zone" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="Filter_Zone">
    <xs:sequence>
        <xs:element name="filter_area" type="Filter_Area" />
        <xs:element name="pattern_width" type="xs:int" />
        <xs:element name="pattern_height" type="xs:int" />
        <xs:element name="filter_width" type="xs:int" />
        <xs:element name="filter_height" type="xs:int" />
        <xs:element name="spectral_range_start_nm" type="xs:double" />
        <xs:element name="spectral_range_end_nm" type="xs:double" />
        <xs:element name="bands" type="Bands" />
    </xs:sequence>
    <xs:attribute name="version" type="xs:int" use="required" fixed="3" />
    <xs:attribute name="layout" type="Filter_Layout_Enum" use="required" />
    <xs:attribute name="index" type="xs:int" use="required" />
</xs:complexType>

<xs:complexType name="Filter_Area">
    <xs:sequence>
        <xs:element name="offset_x" type="xs:int" />
        <xs:element name="offset_y" type="xs:int" />
        <xs:element name="width" type="xs:int" />
        <xs:element name="height" type="xs:int" />
    </xs:sequence>
    <xs:attribute name="version" type="xs:int" use="required" fixed="0"/>
</xs:complexType>

<xs:complexType name="Bands">
    <xs:sequence>
        <xs:element name="band" type="Band" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="Band">
    <xs:sequence>
        <xs:element name="peaks" type="Peaks"/>
        <xs:element name="response" type="Value_List_Double"/>
    </xs:sequence>
    <xs:attribute name="version" type="xs:int" use="required" fixed="4" />
    <xs:attribute name="index" type="xs:int" use="required" />
    <xs:attribute name="selected" type="xs:boolean" use="required" />
</xs:complexType>

<xs:complexType name="Peaks">
    <xs:sequence>
        <xs:element name="peak" type="Peak" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="Peak">
    <xs:sequence>
        <xs:element name="wavelength_nm" type="xs:double" />
        <xs:element name="fwhm_nm" type="xs:double" />
        <xs:element name="QE" type="xs:double" />
        <xs:element name="contribution" type="xs:double" />
        <xs:element name="fit_error" type="xs:double" />
    </xs:sequence>
    <xs:attribute name="version" type="xs:int" use="required" fixed="2" />
    <xs:attribute name="order" type="xs:int" use="required" />
    <xs:attribute name="shape" type="Peak_Shape_Enum" use="required" />
</xs:complexType>


<!-- main object system_info-->
<xs:complexType name="System_Info">
    <xs:sequence>
        <xs:element name="optical_components" type="Optical_Components" minOccurs="0"/>
```

```
            <xs:element name="spectral_correction_info" type="Spectral_Correction_Info"/>
        </xs:sequence>
        <xs:attribute name="version" type="xs:int" use="required" fixed="0" />
    </xs:complexType>

    <xs:complexType name="Optical_Components" mixed="true" >
        <xs:sequence>
            <xs:element name="optical_component" type="Optical_Component" minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="Optical_Component">
        <xs:sequence>
            <xs:element name="type" type="Optical_Component_Type_Enum" />
            <xs:element name="manufacturer" type="xs:string" />
            <xs:element name="part_id" type="xs:string" />
            <xs:element name="tag" type="xs:string" />
            <xs:element name="description" type="xs:string" />
            <xs:element name="measurement" type="xs:string" />
            <xs:element name="transmission_range_start_nm" type="xs:double" />
            <xs:element name="transmission_range_end_nm" type="xs:double" />
            <xs:element name="measured" type="xs:date" />
            <xs:element name="sample_points_nm" type="Value_List_Double"/>
            <xs:element name="response" type="Value_List_Double" />
        </xs:sequence>
        <xs:attribute name="version" type="xs:int" use="required" fixed="2" />
    </xs:complexType>

    <xs:complexType name="Spectral_Correction_Info">
        <xs:sequence>
            <xs:element name="correction_matrices" type="Correction_Matrices" />
        </xs:sequence>
        <xs:attribute name="version" type="xs:int" use="required" fixed="0" />
    </xs:complexType>

    <xs:complexType name="Correction_Matrices">
        <xs:sequence>
            <xs:element name="correction_matrix" type="Correction_Matrix" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="Correction_Matrix">
        <xs:sequence>
            <xs:element name="name" type="xs:string" />
            <xs:element name="algorithm" type="Correction_Matrix_Algorithm_Enum" />
            <xs:element name="algorithm_version" type="xs:string" />
            <xs:element name="type" type="Correction_Matrix_Type_Enum" />
            <xs:element name="minimum_band_energy" type="xs:double" />
            <xs:element name="optical_components" type="Optical_Components" minOccurs="0" />
            <xs:element name="virtual_bands" type="Virtual_Bands" />
        </xs:sequence>
        <xs:attribute name="version" type="xs:int" use="required" fixed="6" />
        <xs:attribute name="created" type="xs:dateTime" use="required" />
    </xs:complexType>

    <xs:complexType name="Virtual_Bands">
        <xs:sequence>
            <xs:element name="virtual_band" type="Virtual_Band" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="Virtual_Band">
        <xs:sequence>
            <xs:element name="wavelength_nm" type="xs:double" />
            <xs:element name="fwhm_nm" type="xs:double" />
            <xs:element name="coefficients" type="Value_List_Double" />
        </xs:sequence>
        <xs:attribute name="version" type="xs:int" use="required" fixed="3" />
    </xs:complexType>
```

14. list of figures

15. list of tables

XIMEA GmbH
Am Mittelhafen 16 • 48155 Münster • Germany • www.ximea.com