Image Processor

Nov 19, 2024

Contents

1 Auto Correlator **3**
- 1.1 Calibration 6
- 1.2 Device Scenes 8
- 1.3 Troubleshooting 9

2 Image Processor **11**
- 2.1 Input to the Device 11
- 2.2 Commands 15
- 2.3 Output of the Device 16

3 Simple Image Processor **25**
- 3.1 Input to the Device 26
- 3.2 Commands 26
- 3.3 Output of the Device 27
- 3.4 Expert Contact 28

4 Image Averager **29**
- 4.1 Input to the Device 29
- 4.2 Commands 30
- 4.3 Output of the Device 30

5 Image Background Subtraction **31**
- 5.1 Input to the Device 31
- 5.2 Commands 32
- 5.3 Output of the Device 32

6 Image Masking **33**
- 6.1 Input to the Device 33
- 6.2 Commands 34
- 6.3 Output of the Device 34

7 Image ROI **35**
- 7.1 Input to the Device 35
- 7.2 Output of the Device 35

8 Normalized Spectrum from ROI **37**
- 8.1 Input to the Device 37

8.2 Output of the Device . . . 38

9 Image Pattern Picker **39**
9.1 Input to the Device . . . 39
9.2 Output of the Device . . . 40

10 Image Picker **41**
10.1 Input to the Device . . . 42
10.2 Output of the Device . . . 43

11 Image to Spectrum **45**
11.1 Input to the Device . . . 45
11.2 Output of the Device . . . 45

12 Beam Shape Coarse **47**
12.1 Commands . . . 47
12.2 Output of the Device . . . 48

13 Image Thumbnail **49**
13.1 Input to the Device . . . 49
13.2 Output of the Device . . . 50

14 Two Peak Finder **51**
14.1 Input to the Device . . . 51
14.2 Commands . . . 51
14.3 Output of the Device . . . 52

15 Expert Contact **53**

16 Indices and tables **55**

The Image Processor devices can be connected to the output channel of a device producing images (usually a camera, or an image processor device).

Incoming data will be sought for images in the `data.image` key.

Contents:

CHAPTER 1

Auto Correlator

The AutoCorrelator device is designed to provide an online determination of the pulse duration using a single-shot auto correlator[1].

The measurement of the time profile of pulses is based on the following principle, graphically displayed in `Fig. %s`. The input beam is sent to a beam-splitter; the two identical beams propagate along two distinct optical paths until they intersect in a non-linear crystal. Here, due to the high-intensity of the beams, a second harmonic beam (SH) is created and its integrated energy is measured by a CCD camera located after the crystal.

The pulse duration of laser pulses can be determined upon measuring the transverse distribution of the energy deposited in the CCD camera. From geometrical considerations in `Fig. %s`, assuming for the incoming beams a rectangular time profile τ_p and uniform transverse intensity profile, it is found that the transverse profile D_z of the second harmonic depends on the pulse duration τ_p of the fundamental beams,

$$D_z = \frac{\tau_p \cdot u}{sin(\phi/2)}$$

$$\tau_p = D_z \cdot \frac{1}{2} \cdot \frac{\Delta t}{\Delta Z_0}$$

where $u = c/n$ and ϕ are the speed of light and the intersection angle of input beams, respectively, in the crystal with refractive index n. The transverse profile D_z is determined from the data accumulated with the CCD camera available in the system. An example of camera image is presented in `Fig. %s`:

The figure shows clearly the deposited energy from the signal of the generated second harmonic beam (central and more intense peak) and of the two fundamental beams (low intensity side signals). The transverse profile D_z is determined as FWHM from the fit to the SH peak.

The angle ϕ cannot be measured with sufficient precision for a reliable extraction of pulse duration τ_p. The way used in the device to determine the pulse duration from the measured transverse profile is presented the calibration section.

The device configuration editor is presented in `Fig. %s`,

The camera device providing the image of the beam profile should be set in the key **input.connectedOutputChannels** of the autocorrelator device. For each camera image the projection along the x-axis is calculated, a fit is performed according to a selectable model (**Beam Shape**) for the time-profile of the pulse, and the peak position and FWHM

[1] RP Photonics Encyclopedia, https://www.rp-photonics.com/autocorrelators.html

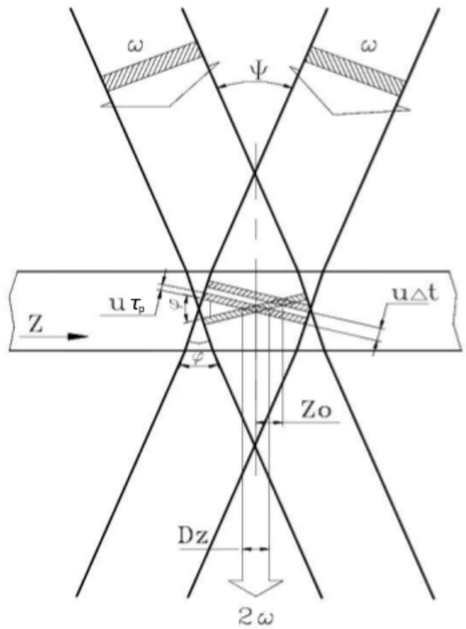

Fig. 1: The diagram describes geometrically the intersection of two identical beams in a crystal and the generation of the second harmonic beam.

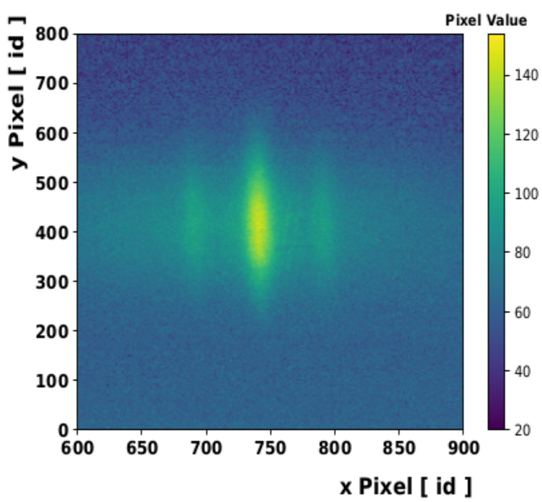

Fig. 2: The fundamental beams and the second harmonic beam are detected in the CCD camera located after the non-linear crystal.

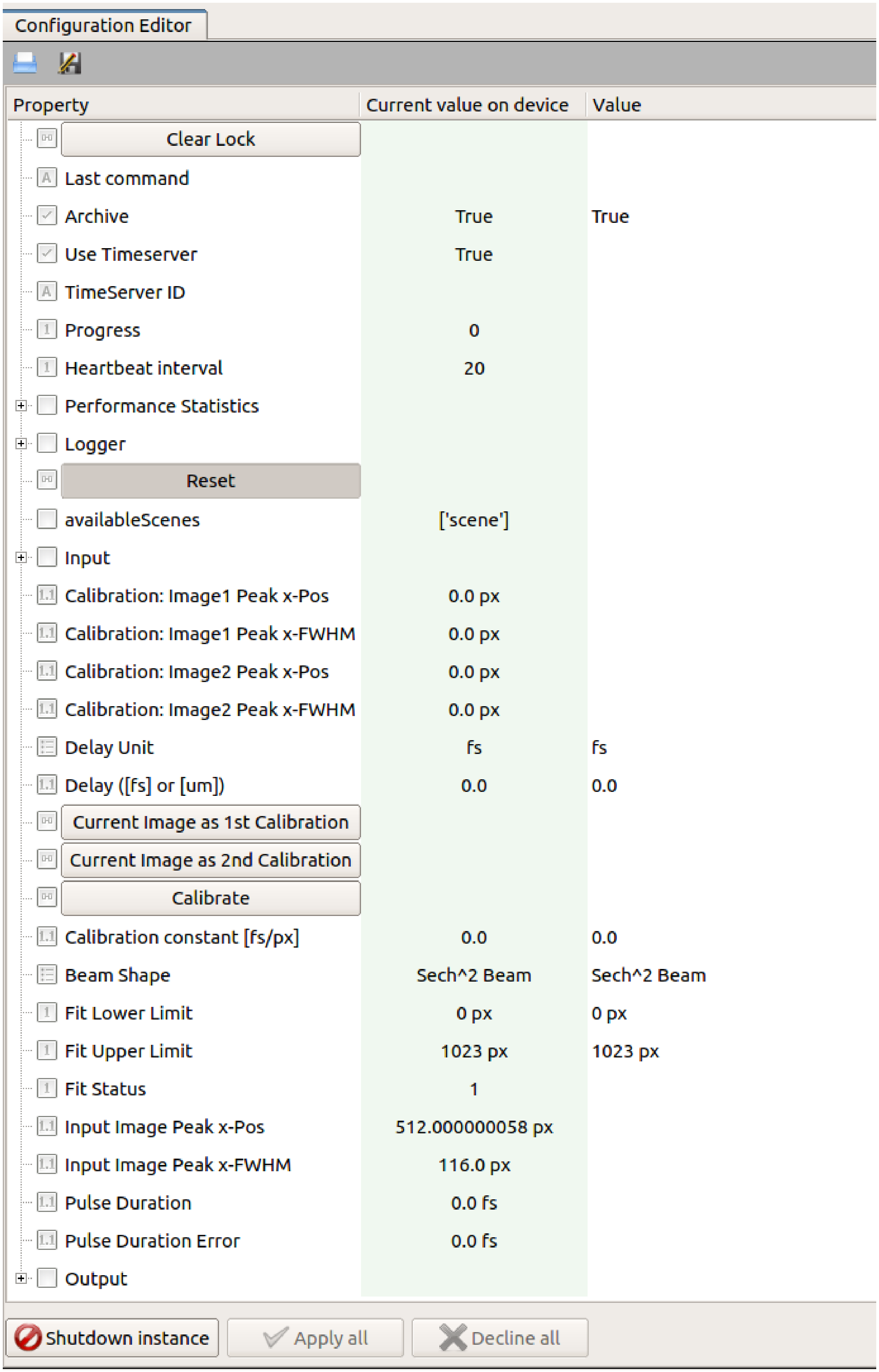

Fig. 3: Configuration Editor of the autocorrelator device.

are determined from the fitting function (**Input Image Peak x-Pos** and **Input Image Peak x-Pos**). The **Fit Error** parameter is an integer flag describing the fit status. If it is equal to 1, 2, 3 or 4, the solution was found, otherwise the solution was not found[2]. The possible fit status values are:

- 0: Improper input parameters were entered,
- 1: The solution converged,
- 2: The number of calls to function has reached default max number,
- 3: Max for relative error is too small, no further improvement in the approximate solution is possible,
- 4: The iteration is not making good progress, as measured by the improvement from the last five Jacobian evaluations,
- 5: The iteration is not making good progress, as measured by the improvement from the last ten iterations,
- 'unknown': "An error occurred.

The result of pulse duration is presented only in case of a solution is found, and the fit status value is lower than four.

1.1 Calibration

To overcome the difficulty in measuring the incident angle ϕ of the primary beams, the following method is applied.

By shifting the mirror stage in the optical delay line, `Fig. %s`, a delay Δt is added between the two input pulses, resulting in a shift ΔZ_0 of the center of SH transverse distribution

$$\Delta Z_0 = \frac{\Delta t \cdot u}{2 \cdot sin(\phi/2)}$$

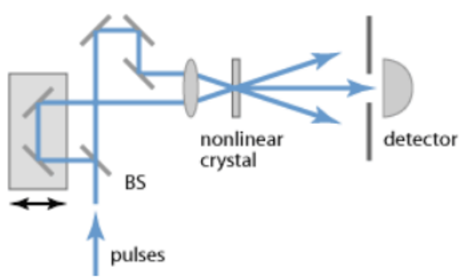

Fig. 4: Setup of an intensity autocorrelator. BS refers to the beam splitter.

Combining equations on transverse profile D_z with shift ΔZ_0 the dependence on the intersection angle ϕ is removed, and the pulse duration can be obtained as

$$\tau_p = D_z \cdot \frac{1}{2} \cdot \frac{\Delta t}{\Delta Z_0}$$

The ratio $K = \frac{\Delta t}{\Delta Z}$ is a calibration factor which allows the conversion of the SH transverse profile (measured in pixel units) to the pulse time profile (measured in femtosecond units).

Its determination with sufficient accuracy is challenging. To overcome this difficulty the following procedure is applied. One of the two optical paths can be varied by pulling or pushing one mirror in the line in a controllable way

[2] Scipy.org, https://github.com/scipy/scipy/blob/master/scipy/optimize/minpack.py

using a micrometer. A change Δl of the micrometer head position results in a pulse delay of $\Delta t = 2\Delta l/c$ and in the shift ΔZ_0. Thus, shifting the SH distribution, as measured in the CCD camera, in two extreme opposite positions (1 & 2) of the sensitive area allows the measurements of calibration factor with a lower relative uncertainty as shown in the steps below:

$$\Delta t = 2\Delta l/c$$

$$\Delta t_1 - \Delta t_2 = 2(\Delta l_1 - \Delta l_2)/c$$

Considering the above expression of τ_p,

$$\Delta t_1 - \Delta t_2 = 2 \cdot \tau_p/D_z(\Delta Z_1 - \Delta Z_2)$$

$$(\Delta l_1 - \Delta l_2)/c = \tau_p/D_z(\Delta Z_1 - \Delta Z_2)$$

resulting in

$$\tau_p = D_z \cdot \frac{1}{2} \cdot (\frac{2}{c} \cdot \frac{\Delta l_1 - \Delta l_2}{\Delta Z_1 - \Delta Z_2})$$

This way, the calibration factor $K = (\frac{2}{c} \cdot \frac{\Delta l_1 - \Delta l_2}{\Delta Z_1 - \Delta Z_2})[\frac{fs}{pxl}]$ can be calculated with a larger relative precision using a reproducible and controllable procedure.

It should be noted that the multiplying factor 1/2 in the above equation results from the initial and non-realistic assumption of a rectangular time profile and uniform transverse intensity profile for the incoming beams. More realistic models for the unknown time shape of initial pulses should be considered. Assuming the Gaussian and hyperbolic secant shapes for the pulse time-profile results in the factors 1/2 and 1/1.54, respectively.

The oscillator pulse duration is then calculated as the mean value of these extracted values, and the contribution from model uncertainty to the global systematical uncertainty can be estimated as half of the maximum deviation between the two calculated values.

The above mentioned calibration steps are handled by the device configuration editor. The user should take care to properly select the fitting region reducing the contribution from the fundamental beams. The fitting window can be optimized configuring the keys **Fit Lower Limit** and **Fit Upper Limit**. Also, attention should be taken in order not to cut the profile tail of the SH beam thus affecting the measurement of the FWHM.

After moving the generated SH beam to one side of the sensitive area of the CCD camera (by properly translating the mirror stage in the optical delay line with the micrometer), by clicking on **Current Image as 1st Calibration** the current values of peak position and FWHM will be set as **Image1 Peak (x)** and **Image1 FWHM (x)**, respectively. Similarly, the second set of calibration parameters are obtained steering the SH profile in the other side of the camera and clicking on **Current Image as 2nd Calibration**.

Once the two calibration images are acquired, the calibration constant K can be calculated by clicking on **Calibrate** after setting

- **Delay Unit** to μm;
- **Delay** to the entire translation of the mirror stage, equivalent to $(\Delta l_1 - \Delta l_2)$. This measurement should be taken by the user;

or, in case the optical delay between the two calibration images was provided already in femtosecond unit, after setting

- **Delay Unit** to fs;
- **Delay** to the time delay.

The extracted **Calibration constant** allows to calculate the pulse duration from the measured FWHM D_z,

$$\tau_p = D_z \cdot \alpha \cdot K,$$

α being the multiplication factor originating from the model assumed for the time-profile of the pulse.

The uncertainty of the pulse duration is preliminary estimated via error propagation by the uncertainty on the fit FWHM, assuming the uncertainty of the calibration constant is negligible and that no correlation between the fit parameters exists.

1.2 Device Scenes

At the moment, one scene is auto-generated by the device.

It can be opened either by right-clicking on the device name, and selecting from the drop up menu the item *Open device scene*, or double-clicking on the device name.

An example of scene is presented in `Fig. %s`:

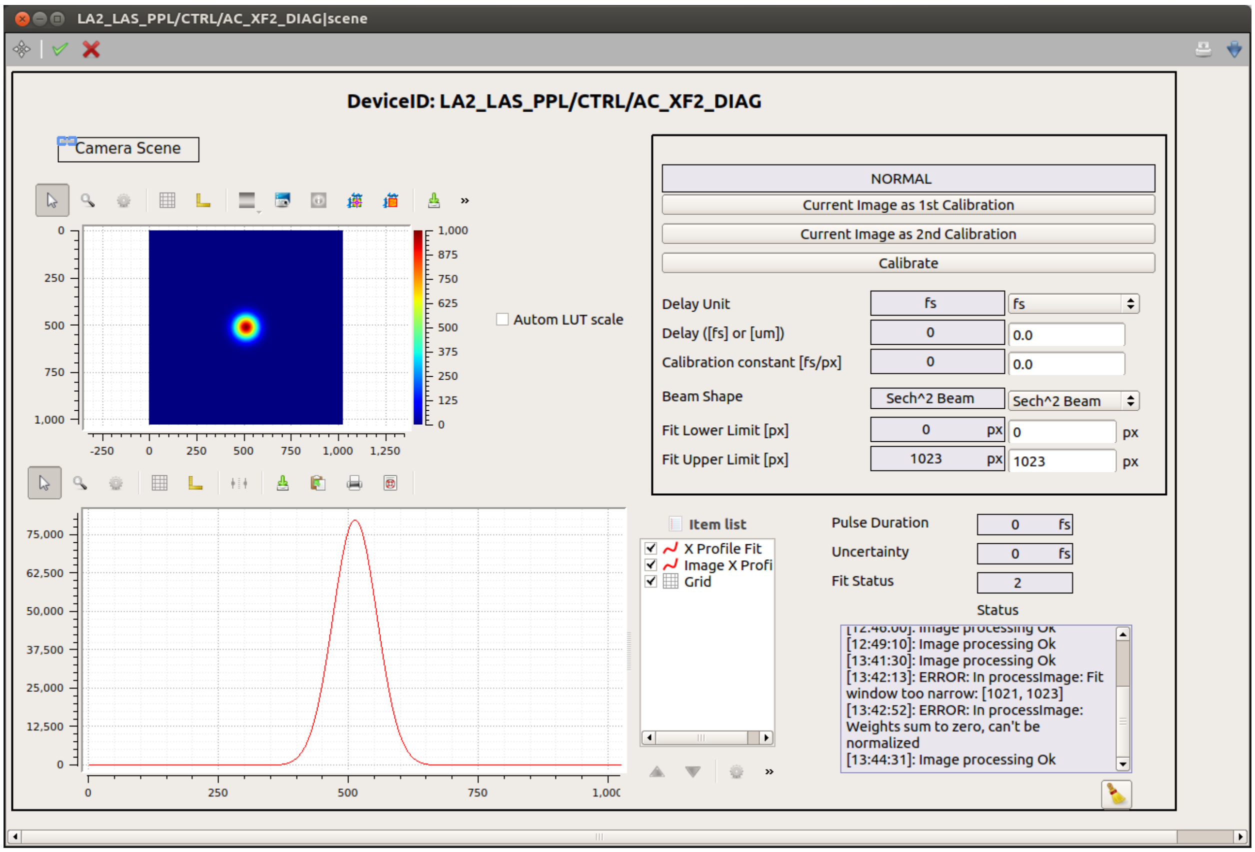

Fig. 5: The scene of the auto-correlator device.

All calibration parameters are available in the upper-right sub-panel. The image x-profile is shown superimposed to the fitting function. To deselect one of the graphs use the item list widget. If not yet visible, this widget can be activated from the drop up menu showing up by right-clicking on the graph.

A log of the device **status** is also provided. Note that only messages appeared after the opening of the scene will be displayed.

A link to the camera auto-generated scene is provided, allowing the user to configure the camera without having to navigate in the project.

1.3 Troubleshooting

Some typical errors have been identified up to now:

- In case the camera device is not instantiated or it is stopped the peak position and FWHM should be null, and no calculation of the pulse duration can be performed;
- In case no calibration constant is provided, either inserted by the user (if previously known) or by following the calibration procedure described in the text, the pulse duration is not calculated;
- In case the calibration constant is inserted by the user, and the results appear to be very different from what expected, the value used might describe no more the current optical setup of the autocorrelator device. A new calibration measurement could be performed;
- In case the uncertainty arising from the fit procedure is relative large, likely the model used in the fit is not appropriate:

  – try to use a different available model;

  – try to optimize the fitting region;

  – verify that the tails of the second harmonic beam are well within the fitting area;
- In case no available model describes correctly the data, an optimization of the optical line setup could be attempted.

CHAPTER 2

Image Processor

The Image Processor device can provide for each incoming image (2D) or spectrum (1D):

- the minimum, maximum and mean pixel value;
- the frequency distribution of pixel values;
- the image integrals in x and y directions (only for 2D images);
- the centre-of-mass and standard deviation;
- gaussian fit parameters for the x and y integrals (the latter only for 2D (images);
- gaussian fit parameters for the 2D image;
- pixel values integral over a region.

Each feature can be enabled or disabled by using the boolean properties listed in the *General Settings* section.

General settings of the Image Processor are described in the *Enabling Features* section.

2.1 Input to the Device

2.1.1 General Settings

The following properties affect all the algorithms ran in the device.

Property key	Description
imagePath	The key where the image will be looked for in the input data. The default value - `data.image` - is usually appropriate when the input channel is connected to imagers.
filterImagesByThreshold	If `filterImagesByThreshold` is `True`, only images with maximum pixel value exceeding `imageThreshold` will be processed. Others will be discarded.
imageThreshold	The threshold for processing an image. See above.
absolutePositions	If `True`, the centre-of-mass and fit results will take into account the current settings for ROI and binning.
subtractBkgImage	Subtract the loaded background image.
subtractImagePedestal	Subtract the image pedestal (i.e. `image = image - image.min()`). This is done after background subtraction.

2.1.2 Enabling Features

The different algorithms available can be enabled by setting the following boolean parameters.

Property key	Description
doMinMaxMean	Get the following information from the pixels: min, max, mean value.
doBinCount	Calculate the frequency distribution of pixel values. The distribution will be available in `data.imgBinCount`.
doXYSum	Integrate the image along the x- and y-axes. The distributions will be available in `data.imgX` and `data.imgY`.
doCOfM	Calculate centre-of-mass and widths.
do1DFit	Perform a 1D gaussian fit of the x- and y-distributions.
do2DFit	Perform a 2D gaussian fits. Be careful: It can be slow!
doIntegration	Integrate the pixel values in the specified region.

2.1.3 Options for Centre-of-Mass

The user can define a range for the centre-of-mass calculation, and a pixel threshold to discard background pixels. More details are given in the table:

Property key	Description
comRange	The range to be used for the centre-of-mass calculation. Can be the full range, or a user-defined one.
userDefinedRange	The user-defined range for centre-of-mass, gaussian fit(s) and integrales along the x & y axes.
absThreshold	Pixels below this threshold will not be used for the centre-of-mass calculation. If greater than 0, the relative threshold will not be used.
threshold	Pixels below this relative threshold (fraction of the highest value) will not be used for the centre-of-mass calculation. It will only be applied if no absolute threshold is set.

2.1.4 Options for Gaussian Fit

The Gaussian fit is done by using the `fitGauss` and `fitGauss2DRot` functions available in the image processing package.

Initial parameters fit are calculated by the `peakParametersEval` function in the imageProcessing package, when the "raw peak" option is choosen.

The user can define the range used for the Gaussian fit, enable a 1st order polynomial, define which initial fit parameters shall be used, enable rotation angle for the 2D Gaussian fit.

More details are given in the table:

Property key	Description
pixelSize	The pixel size. It will be used when evaluating the beam size.
fitRange	The range to be used for fitting. Can be the full range, an auto-determined, or the user-defined one.
rangeForAuto	The automatic range for 'auto' mode (in standard deviations).
userDefinedRange	The user-defined range.
enablePolynomial	Add a 1st order polynomial term (ramp) to gaussian fits.
gauss1dStartValues	Selects how 1d gaussian fit starting values are evaluated. The options are: last fit result, raw peak.
doGaussRotation	Allow the 2D gaussian to be rotated.

2.1.5 Options for Integration

The user can define the region to be integrated over.

Property key	Description
integrationRegion	The region to be integrated over.

2.2 Commands

The user can select the current image as background image.

Slot key	Description
useAsBackgroundImage	Use the current image as background image.

2.3 Output of the Device

2.3.1 General properties

Property key	Description
frameRate	The rate of incoming images. It is refreshed once per second.
imageWidth	The width of the incoming image.
imageOffsetX	If the incoming image has a ROI, this represents the X position of the top-left corner.
imageBinningX	The image binning in the X direction.
imageHeight	The height of the incoming image.
imageOffsetY	If the incoming image has a ROI, this represents the Y position of the top-left corner.
imageBinningY	The image binning in the Y direction.
minPxValue	The minimum image pixel value.
maxPxValue	The maximum image pixel value.
meanPxValue	The mean image pixel value.

2.3.2 Execution Time

The time spent in each part of the image processing is calculated and displayed in the device. The values are refreshed once per second.

Property key	Description
minMaxMeanTime	Time spent for evaluating min, max, mean pixel value.
binCountTime	Time spent for calculating the frequency distribution of pixel values.
subtractBkgImageTime	Time spent in subtracting the background image.
subtractPedestalTime	Time spent in subtracting the image pedestal.
xYSumTime	Time spent in integrating the image in X and Y.
cOfMTime	Time spent in evaluating the centre-of-mass.
xFitTime	Time spent in 1D Gaussian fit of the X distribution.
yFitTime	Time spent in 1D Gaussian fit of the Y distribution.
fitTime	Time spent in 2D Gaussian fit of the image.
integrationTime	Time spent in integrating over a region.

2.3.3 Centre-of-Mass

Property key	Description
x0	X position of the centre-of-mass.
sx	Standard deviation in X of the centre-of-mass.
y0	Y position of the centre-of-mass.
sy	Standard deviation in Y of the centre-of-mass.

2.3.4 Gaussian Fit

By enabling the 1D fits, the image will be first integrated along Y- and X- directions, in order to give a 1D distribution. These distributions will be then fitted with a Gaussian.

Property key	Description
xFitSuccess	1D Gaussian fit success for the X distribution (1-4 if fit converged).
ax1d	Amplitude `Ax` from 1D fit.
x01d	`x0` peak position from 1D fit.
ex01d	Uncertainty on `x0` estimation.
sx1d	Standard deviation on `x0` from 1D fit.
esx1d	Uncertainty on standard deviation estimation.
beamWidth1d	Beam width from 1D Fit. Defined as 4x `sx1d`.
yFitSuccess	1D Gaussian fit success for the Y distribution (1-4 if fit converged).
ay1d	Amplitude `Ay` from 1D fit.
y01d	`y0` peak position from 1D fit.
ey01d	Uncertainty on `y0` estimation.
sy1d	Standard deviation on `y0` from 1D fit.
esy1d	Uncertainty on standard deviation estimation.
beamHeight1d	Beam height from 1D Fit. Defined as 4x `sy1d`.

By enabling the 2D fit, the 2D pixel distribution will be fitted. Be careful, for large images it could be quite slow, in particular if you enable rotation angle!

Property key	Description
fitSuccess	2D Gaussian fit success (1-4 if fit converged).
a2d	Amplitude from 2D fit.
x02d	`x0` peak position from 2D fit.
ex02d	Uncertainty on `x0` estimation.
sx2d	Standard deviation on `x0` from 2D fit.
esx2d	Uncertainty on standard deviation estimation.
beamWidth2d	Beam width from 2D Fit. Defined as 4x `sx2d`.
y02d	`y0` peak position from 2D fit.
ey02d	Uncertainty on `y0` estimation.
sy2d	Standard deviation on `y0` from 2D fit.
esy2d	Uncertainty on standard deviation estimation.
beamHeight2d	Beam height from 2D Fit. Defined as 4x `sy2d`.
theta2d	Rotation angle from 2D fit.
etheta2d	Uncertainty on rotation angle estimation.

2.3.5 Integration

Property key	Description
regionIntegral	Integral of pixel value over the specified region.
regionMean	Mean pixel value over the specified region.

2.3.6 Other Outputs

The following vector properties are available in the output channel named *output*.

Property key	Description
data.imgBinCount	Distribution of the image pixel counts.
data.imgX	Image integral along the Y-axis.
data.imgY	Image integral along the X-axis.

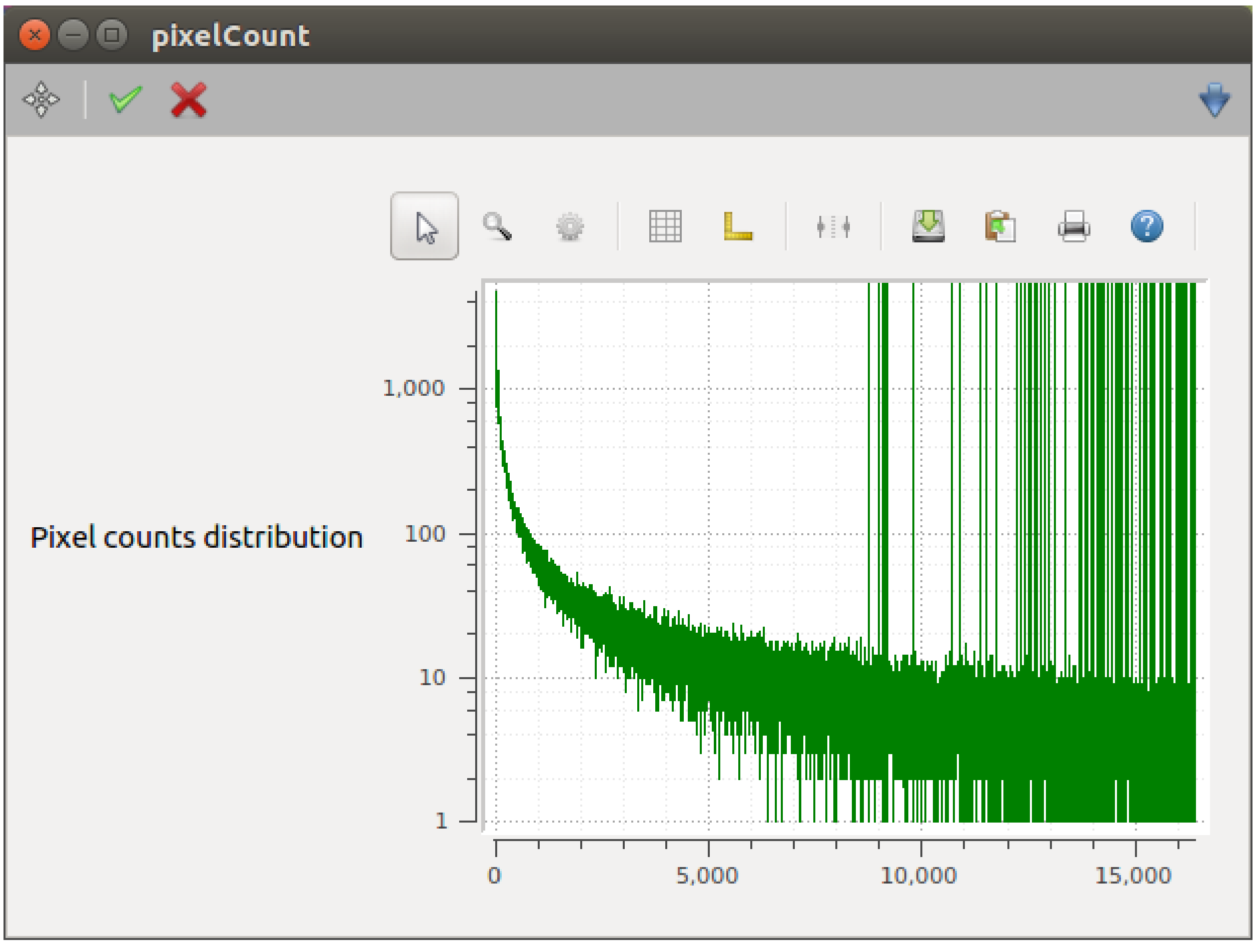

Fig. 1: An example of pixel count distribution.

CHAPTER 3

Simple Image Processor

The Simple Image Processor device can be connected to the output channel of a device producing images (usually a camera, or an image processor device).

Incoming data will be sought for images in the `data.image` key.

The Simple Image Processor device can provide for each incoming image:

- the maximum pixel value;
- gaussian fit parameters for the x and y integrals.

The settings of the Simple Image Processor are described in the next section.

3.1 Input to the Device

Property key	Description
pixelSize	The pixel size, to be used for converting the fit's standard deviation to FWHM.
imageThreshold	The threshold for doing processing. Only images having maximum pixel value above this threshold will be processed.
subtractImagePedestal	Set to *True*, to subtract the image pedestal (i.e. image = image - image.min()) before centre-of-mass and Gaussian fit.
thresholdType	Defines whether an absolute or relative thresholding is used in the calculations.
pixelThreshold	If thresholdType is set to 'absolute', pixels below this threshold will be set to 0 in the processing of images. If it is set to 'relative', pixels below this fraction of the maximum pixel value will be set to zero. If it is set to None, no thresholding will occur.

3.2 Commands

Slot key	Description
reset	Resets the processor output values.

3.3 Output of the Device

3.3.1 General properties

Property key	Description
frameRate	The actual frame rate.
imageSizeX	The image width.
imageSizeY	The image height.
offsetX	The image offset in X direction, i.e. the X position of its top-left corner.
offsetY	The image offset in Y direction, i.e. the Y position of its top-left corner.
binningX	The image binning in X direction.
binningY	The image binning in Y direction.

3.3.2 Gaussian Fit

Property key	Description
success	Success boolean whether the image processing was successful or not.
maxPxValue	Maximum pixel value.
amplitudeX, amplitudeY	Amplitude from Gaussian fit.
positionX, positionY	Beam position from Gaussian fit.
sigmaX, sigmaY	Standard deviation from Gaussian fit.
fwhmX, fwhmY	FWHM obtained from standard deviation.
errSigmaX, errSigmaY	Uncertainty on position from Gaussian fit.

3.4 Expert Contact

- Dennis Goeries <dennis.goeries@xfel.eu>
- Andrea Parenti <andrea.parenti@xfel.eu>

CHAPTER 4

Image Averager

The *ImageAverager* device can perform a running average, or the standard one, of the incoming images. Its settings are described in the *Input to the Device* section.

4.1 Input to the Device

Property key	Description
nImages	The number of images to be averaged.
runningAverage	If `True`, a moving average is calculated, otherwise the standard average.
runningAvgMethod	The algorithm used to calculate the running average it can be either `ExactRunningAverage` to use a simple moving average, or `ExponentialRunningAverage` to use an exp moving average.

4.2 Commands

Slot key	Description
resetAverage	Resets all temporary variables used for averaging.

4.3 Output of the Device

Property key	Description
inFrameRate	The rate of incoming images. It is refreshed once per second.
outFrameRate	The rate of averaged images written to the output channel. It is refreshed once per second.
ppOutput	The output channel for GUI and pipelines. The averaged imaged can be found in `data.image`.
daqOutput	The output channel for DAQ - with reshaped image.

CHAPTER 5

Image Background Subtraction

The *ImageBackgroundSubtraction* device can subtract a background image from the incoming one. Its settings are described in the *Input to the Device* section.

5.1 Input to the Device

Property key	Description
disable	Disable background subtraction.
imageFilename	The full filename to the background image. File format must be 'npy', 'raw' or TIFF.

5.2 Commands

Slot key	Description
resetBackgroundImage	Reset background image.
save	Save to file the current image.
load	Load a background image from file.
useAsBackgroundImage	Use the current image as background image.
reset	Reset error count.

5.3 Output of the Device

Property key	Description
frameRate	The rate of incoming images. It is refreshed once per second.
errorCount	Number of errors.
ppOutput	The output channel for GUI and pipelines. The background subtracted imaged can be found in `data.image`.
daqOutput	The output channel for DAQ - with reshaped image.

CHAPTER 6

Image Masking

The *ImageApplyMask* device applies a mask to the incoming image, and writes the masked image to an output channel.

6.1 Input to the Device

Property key	Description
disable	No mask will be applied, if set to `True`.
maskType	The mask type: rectangular or arbitrary (loaded from file).
x1x2y1y2	The rectangular selected region: x1, x2, y1, y2.
maskFilename	The full path to the mask file. File format must be *npy*, *raw* or *TIFF*. Pixel value will be set to 0, where mask is <= 0.

6.2 Commands

Slot key	Description
resetMask	Discard the loaded mask.
loadMask	Load the mask from a file.

6.3 Output of the Device

Property key	Description
frameRate	The rate of incoming images. It is refreshed once per second.
output	The output channel for GUI and pipelines. The masked image can be found in `data.image`.

CHAPTER 7

Image ROI

The *ImageApplyROI* device applies a ROI to the incoming image, and writes the sub-image to an output channel.

7.1 Input to the Device

Property key	Description
disable	No ROI will be applied, if set to `True`.
roi	The user-defined region of interest (ROI), specified as [lowX, highX, lowY, highY].

7.2 Output of the Device

Property key	Description
frameRate	The rate of incoming images. It is refreshed once per second.
output	The output channel for GUI and pipelines. The ROI-ed image can be found in `data.image`.

CHAPTER 8

Normalized Spectrum from ROI

The *ImageNormRoi* device is used to calculate an inline normalized spectrum from an image. In order to compute the spectrum, the operator has to define a data region of interest (ROI) and a normalization ROI from the incoming image. Both regions of interest are created with the same size (`roiSize`) and the positions can be defined by `dataRoiPosition` and `normRoiPosition`, respectively. The normalization ROI is then subtracted from the pixel values of the data region of interest and the result is finally integrated along the Y direction to retrieve the spectrum.

8.1 Input to the Device

Property key	Description
roiSize	The size of the user-defined ROI, specified as [width_roi, height_roi].
dataRoiPosition	The position of the user-defined data ROI of the image, specified as [x, y]. Coordinates are counted from top-left corner!
normRoiPosition	The position of the user-defined ROI to normalize the image, specified as [x, y].

8.2 Output of the Device

Property key	Description
output	It will contain the calculated spectrum, in the `data.spectrum` key.
spectrumIntegral	The sum of the spectrum values.

CHAPTER 9

Image Pattern Picker

The aim of this device is to filter input images according to their train IDs.

The image pattern picker has two nodes (`chan_1` and `chan_2`); each of them contains an input channel that can be connected to an output channel to receive an image stream (e.g. from a camera).

The input image has to be found in the `data.image` element. If its `trainId` fulfills a given condition (see next Section), it will be forwarded to the output channel in the same node.

9.1 Input to the Device

Property key	Description
nBunchPatterns	Number of bunch patterns.
patternOffset	The image will be forwarded to the output if its `trainId` satisfies the following relation: `(trainId % nBunchPatterns) == patternOffset`.

9.2 Output of the Device

Property key	Description
inFrameRate	The rate of incoming images. It is refreshed once per second.
outFrameRate	The rate of averaged images written to the output channel. It is refreshed once per second.
output	The output channel. The forwarded images can be found in `data.image`.

CHAPTER 10

Image Picker

This device has two input channels (`inputImage` and `inputTrainid`).

- `inputImage` expects an image stream (e.g. from a camera);
- `inputTrainId` is used to get the timestamps. Its data content is ignored, as only timestamp is relevant.

Images whose `trainId` equals `inputTrainId + trainIdOffset` are forwarded to an output channel, while others are discarded.

10.1 Input to the Device

Property key	Description
isDisabled	When disabled, all images received in input are forwarded to output channel.
imgBufferSize	Number of images to be kept waiting for a matching train ID.
trainIdOffset	Train ID Offset. If positive: output image train ID is greater than input train ID (delay). If negative: output image train ID is lower than input train (advance).
trainIdBufferSize	Number of train IDs to be kept waiting for an image with matching train ID.
inputImage	The input channel for the image stream.
inputTrainId	The input channel for train IDs.

10.2 Output of the Device

Property key	Description
inFrameRate	The rate of incoming images. It is refreshed once per second.
outFrameRate	The rate of averaged images written to the output channel. It is refreshed once per second.
ppOutput	The output channel for GUI and pipelines. The averaged imaged can be found in `data.image`.
daqOutput	The output channel for DAQ - with reshaped image.

CHAPTER 11

Image to Spectrum

The *ImageToSpectrum* device is used to calculate an inline spectrum from an image. In order to compute the spectrum, the operator has to define a region of interest (ROI) from the incoming image. After the selection of the ROI, the image is integrated along the Y direction to retrieve the spectrum.

11.1 Input to the Device

Property key	Description
roi	The user-defined region of interest, specified as [lowX, highX]. [0, 0] will be interpreted as 'whole range'.

11.2 Output of the Device

Property key	Description
output	It will contain the calculated spectrum, in the `data.spectrum` key.
spectrumIntegral	The sum of the spectrum values.

CHAPTER 12

Beam Shape Coarse

The *BeamShapeCoarse* device integrates the incoming images in Y and X directions, then finds the position of the peak and the beam size on such integrals.

Position and size of the beam are calculated with the *peakParametersEval* function from the image processing package, thus the evaluated values make sense only if the peak has a single maximum. Also noise (ripple) may affect the result.

12.1 Commands

Slot key	Description
resetError	Resets the error state.

12.2 Output of the Device

Property key	Description
x0	X coordinate of the maximum intensity pixel.
y0	Y coordinate of the maximum intensity pixel.
fwhmX	Full Width at Half Maximum for X projection, A.K.A. beam width.
fwhmY	Full Width at Half Maximum for Y projection, A.K.A. beam height.
frameRate	Rate of processed images. It is refreshed once per second.

CHAPTER 13

Image Thumbnail

The *ImageThumbnail* device is meant to reduce the input image for preview purposes. It expects an image in input.

It lets the user specify the size of a canvas where the output thumbnail image must fit. It outputs the image downscaled to fit in the specified canvas. Downscaled image is obtained by means of the *thumbnail* function from the image processing package.

13.1 Input to the Device

Property key	Description
thumbCanvas	Shape of the canvas where thumbanail must fit [Y, X]
resample	If `True` binned pixel are averaged. Set to *False* to spare CPU load, set to *True* to avoid aliasing.

13.2 Output of the Device

Property key	Description
frameRate	The rate of incoming and outgoing images. It is refreshed once per second.
ppOutput	The output channel for GUI and pipelines. Thumbnail image can be found in `data.image`.
daqOutput	The output channel for DAQ - with reshaped image.

CHAPTER 14

Two Peak Finder

The *TwoPeakFinder* device will integrate the input image in the vertical direction, then find two peaks, one left and one right from the *zero_point*.

14.1 Input to the Device

Property key	Description
zeroPoint	The device will try to find a peak left, and a peak right, from this point.
threshold	TODO - currently unused.
roi	The user-defined region of interest (ROI), specified as [lowX, highX]. [0, 0] will be interpreted as 'whole range'.

14.2 Commands

Slot key	Description
reset	Reset error count.

14.3 Output of the Device

Property key	Description
frameRate	The rate of incoming and outgoing images. It is refreshed once per second.
errorCount	Number of errors.
peak1Value	Amplitude of the 1st peak.
peak1Position	Position of the 1st peak.
peak1Fwhm	FWHM of the 1st peak.
peak2Value	Amplitude of the 2nd peak.
peak2Position	Position of the 2nd peak.
peak2Fwhm	FWHM of the 2nd peak.

CHAPTER 15

Expert Contact

- Andrea Parenti <andrea.parenti@xfel.eu>

CHAPTER 16

Indices and tables

- genindex
- modindex
- search