histogram

March 9, 2017

1 Histogram vs Histogram_lut

```
In [ ]: import numpy
        from silx.math.histogram import Histogramnd, HistogramndLut
        from silx.gui.plot import Plot1D, Plot2D
        %gui qt
```

This function create some data with noise.

```
In [ ]: def createDataSet():
            shape = (400, 400)
            xcenter = shape[0]/2
            ycenter = shape[1]/2
            t = numpy.zeros(shape)
            y, x=numpy.ogrid[:t.shape[0], :t.shape[1]]
            r=1.0+numpy.sin(numpy.sqrt((x-xcenter)**2+(y-ycenter)**2)/8.0)
            return r + numpy.random.rand(shape[0], shape[1])

        data = createDataSet()
```

Simple display of the fist element of the list

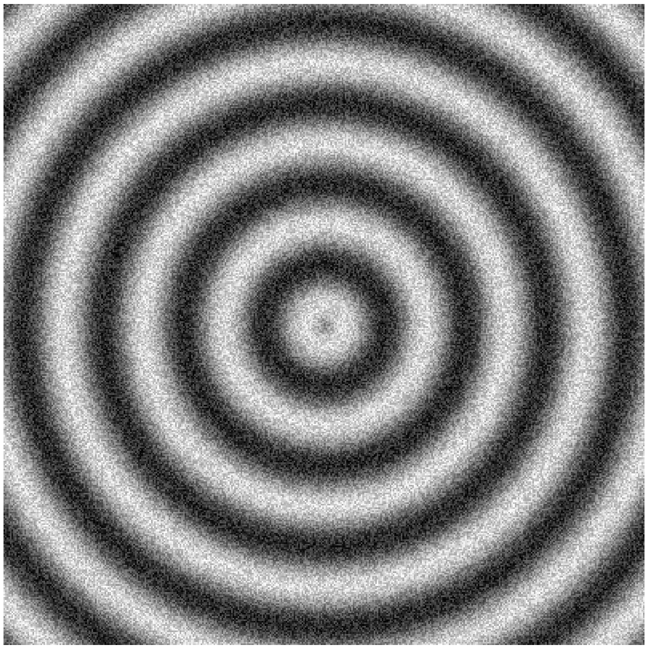
```
In [ ]: p = Plot2D()
        p.addImage(legend='dataExample', data=data)
        p.show()
```

1.1 Exercise : use Histogramnd to compute azimutal integration

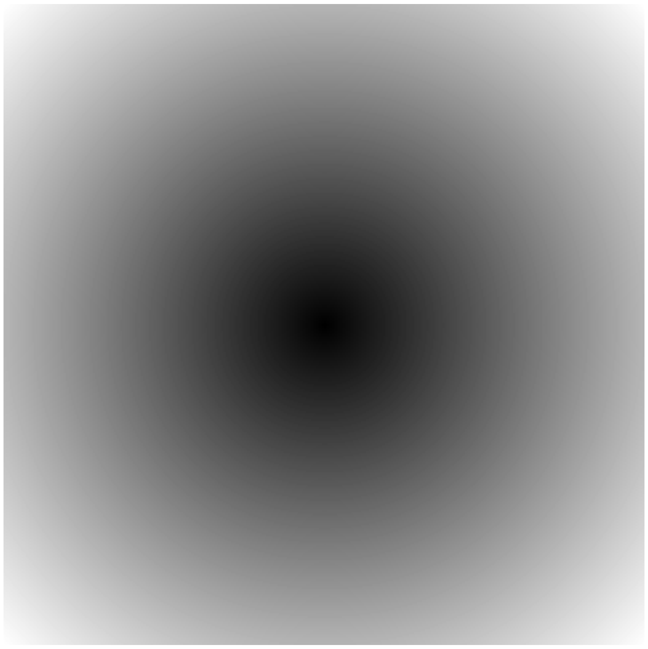
1.1.1 we compute raddi to center for each pixel

```
In [ ]: def computeradius(data):
            xcenter=data.shape[0]/2
            ycenter=data.shape[1]/2
            y, x=numpy.ogrid[:data.shape[0], :data.shape[1]]
            r=numpy.sqrt((x-xcenter)**2+(y-ycenter)**2)
            return r
```

```
In [ ]: radii = computeradius(data)
        # TODO : plot radii data into a Plot2D widget
```



input data



distance pixel-image center

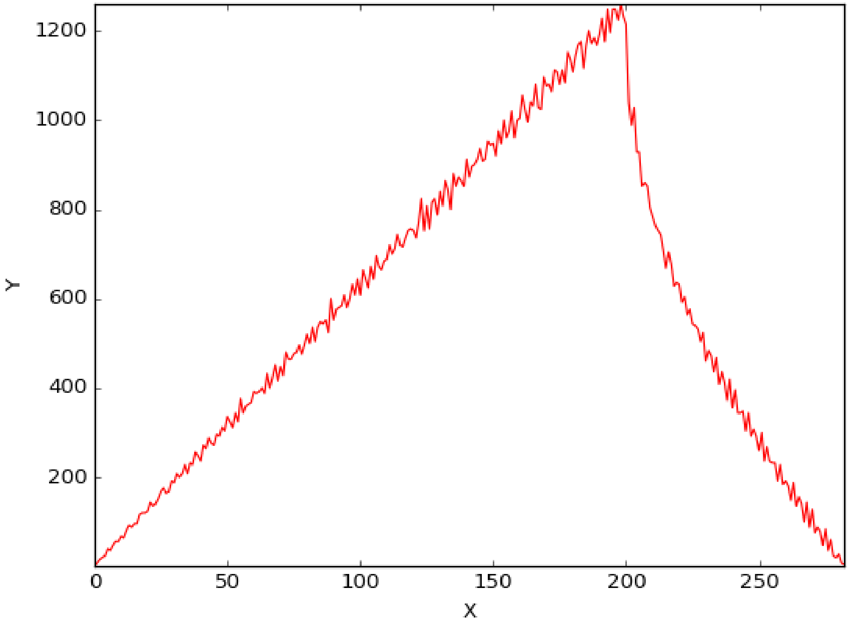
1.1.2 plot the histogram of the radii

documentation :

- http://pythonhosted.org/silx/modules/math/histogram.html

```
In [ ]: nb_bins = int(numpy.ceil(radii.max()))
        histo_range = [0, nb_bins]
        # TODO : compute the histogram of the radii distribution
```

```
In [ ]: # TODO : plot the histogram into a Plot1D widget
```



distance pixel-image center

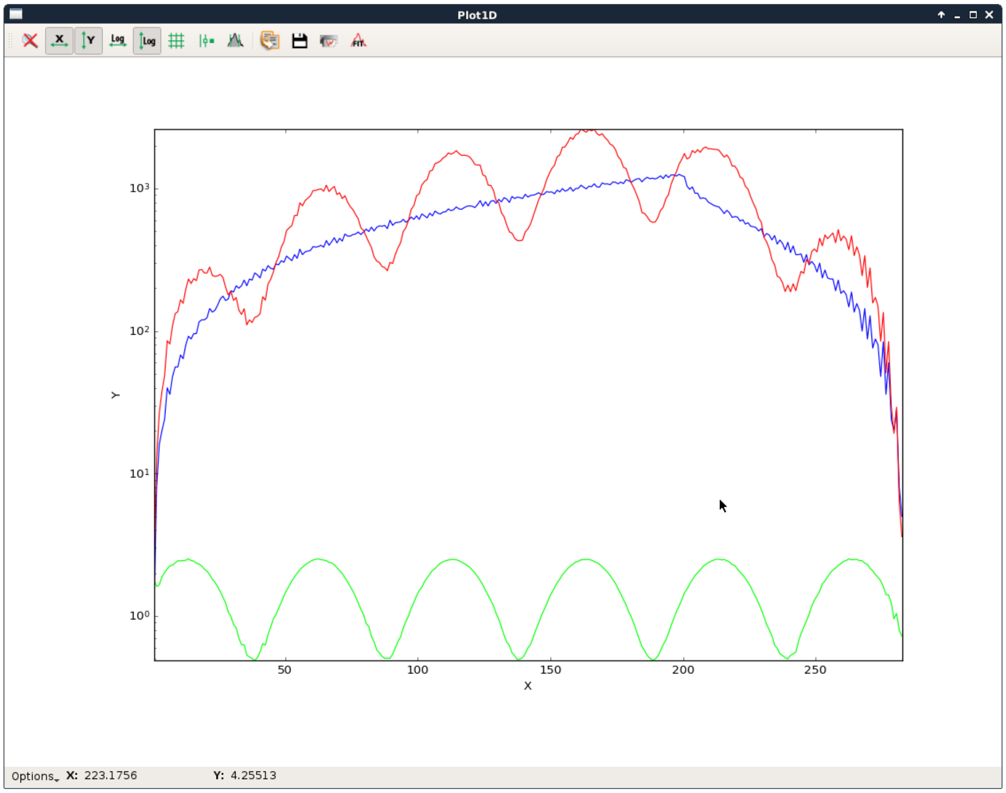
1.1.3 compute azimutal integration

goal : get the mean contribution of each pixels for each radius
step 1 : get the contribution of each pixels for each radius

```
In [ ]: nb_bins = int(numpy.ceil(radii.max()))
        histo_range = [0, nb_bins]
        # TODO : compute the weighted histogram of the contribution of pixel for
        # each radius
```

step 2 : get the mean and plot it

```
In [ ]: # TODO : display the histogram, the weighted_histogram and the normalized
        # histogram (weighted_histogram/histogram) into a Plot1D widget
```



Options X: 223.1756 Y: 4.25513

integration

2 Exercice : compute the azimutal integration over n images

we want to reproduced the same action but over a stack of image : - pixel distance two the center is not evolving - only pixel values are

```
In [ ]: dataset = [ createDataSet() for i in range(10) ]
```

2.1 First way : using Histogramnd

```
In [ ]: def computeDataSetHisto():
            # TODO : create the function returning the histogram accumulating
            # the contribution of pixels for all images in the dataset using
            # Histogramnd Class
            pass
```

```
In [ ]: # plot It
        plotDataSetHistoNd = Plot1D()
        histogramDS = computeDataSetHisto()
        binscenter=(histogramDS.edges[0][1:] + histogramDS.edges[0][0:-1]) / 2.0
        normalization=histogramDS.weighted_histo/histogramDS.histo
        plotDataSetHistoNd.addCurve(x=binscenter, y=normalization, color='red')
        plotDataSetHistoNd.show()
```

2.2 second way : using HistogramndLut

```
In [ ]: def computeDataSetHistoLut():
            # TODO : create the function returning the histogram accumulating
            # the contribution of pixels for all images in the dataset using
            # HistogramndLut Class
            pass
```

```
In [ ]: # plot It
        plotDataSetHistoLut = Plot1D()
        histogramLut = computeDataSetHistoLut()
        normalization=histogramLut.weighted_histo()/histogramDS.histo
        plotDataSetHistoLut.addCurve(binscenter, y=normalization, color='red')
        plotDataSetHistoLut.show()
```

2.3 Compare results

```
In [ ]: numpy.array_equal(histogramLut.weighted_histo(), histogramDS.weighted_histo
```

2.4 Compare execution time

```
In [ ]: %timeit computeDataSetHisto()
```

```
In [ ]: %timeit computeDataSetHistoLut()
```