ImageProcessing Documentation

Release trunk

A. Parenti

Apr 29, 2025

Contents

Contents:

CHAPTER 1

The Karabo's imageProcessing package

1.1 Introduction

The imageProcessing package provides two python submodules, image_processing.image_processing and image_processing.image_running_mean.

In image_processing.image_processing a number of functions is provided, which can be used to do some basic processing on a numpy.ndarray.

In image_processing.image_running_mean the ImageRunningMean class is provided, which can be used to calculate a simple moving average or a cumulative moving average (please find here the definition) of a numpy.ndarray.

1.2 The image_processing submodule

The functions in image_processing submodule can be imported with:

```
from image_processing.image_processing import *
```

Many of the functions will by default modify the input image: to avoid that use the copy=False option.

The following functions are available:

imagePixelValueFrequencies(*image*)
    Call internally numpy.bincount, and return the one dimensional ndarray of pixel value frequencies. Input image must have integer dtype.

imageSetThreshold(*image*, *threshold*, *copy=False*)
    Pixels below threshold are set to zero.

imageSubtractBackground(*image*, *background*, *copy=False*)
    Background is subtracted from image. Beware the image data type: It should be signed since subtraction can result in negative values!

imageApplyMask (*image*, *mask*, *copy=False*)
mask is applied to the image. Mask and image must have the same shape. Image pixels are masked out, where mask values are equal or below 0.

imageSelectRegion (*image*, *x1*, *x2*, *y1*, *y2*, *copy=False*)
Everything outside the rectangular region is set to zero.

imageSumAlongY (*image*)
Image is integrated along the Y axis.

imageSumAlongX (*image*)
Image is integrated along the X axis.

imageCentreOfMass (*image*)
Return centre-of-mass and standard deviation for an image (1D or 2D). If the image is 1D it returns (x0, sx); if the image is 2D it returns (x0, y0, sx, sy). For the calculations numpy.average and numpy.sqrt are used.

fitGauss (*image*, *p0=None*, *enablePolynomial=False*)
Return the gaussian fit parameters of an image (1D or 2D). An initial estimation of the parameters (p0) can be provided. Additionally a first order polynomial a*x + b*y + c can be added to the gaussian. Returned values are (A, x0, sx, covariance, error) or (A, x0, sx, a, c, covariance, error) in case of 1D, (A, x0, y0, sx, sy, covariance, error) or (A, x0, y0, sx, sy, a, b, c, covariance, error) in case of 2D. The fit is done with the help of the scipy.optimize.leastsq function, with full_output option.

fitGauss2DRot (*image*, *p0=None*, *enablePolynomial=False*)
Return the gaussian fit parameters of a 2D image, including rotation. An initial estimation of the parameters (p0) can be provided. Additionally a first order polynomial a*x + b*y + c can be added to the gaussian. Returned values are (A, x0, y0, sx, sy, theta, covariance, error) or (A, x0, y0, sx, sy, theta, a, b, c, covariance, error). The fit is done with the help of the scipy.optimize.leastsq function, with full_output option.

fitSech2 (*image*, *p0=None*, *enablePolynomial=False*)
Return squared hyperbolic secant fit parameters of a 1D image. An initial estimation of the parameters (p0) can be provided. Additionally add a 1st order polynomial a*x + b*y +c.

peakParametersEval (*img*)
Evaluate peak parameters (maxValue, maxPosition, FWHM) for 1D distributions. Parameters are calculated from raw data with no assumption on peak shape, but having a single maximum.

thumbnail (*image*, *canvas*, *resample=False*)
Input image is downscaled by an integer factor to fit in specified canvas, keeping the original x-y ratio. If necessary, original image is padded (with 0 if resample == False, with edge values if True) if this is necessary to keep the ratio. If the original image already fits in the rectangle, it's returned unchanged.

1.3 The ImageRunningMean class

This image_processing.image_running_mean submodule provide the ImageRunningMean class:

```
from image_processing.image_running_mean import ImageRunningMean
```

The available methods for the class are:

append (*image*, *maxlen=None*)
Append an image to the queue and update the running mean. By default the cumulative moving average is calculated, if you want the simple moving average you have to use the parameter maxlen to pop the excess images. The method will raise a ValueError if the image has not the same shape as the other ones in the queue.

popleft ()
Pop an image from the queue, and update the running mean.

clear()
    Clear the queue and reset the running mean.

recalculate()
    Recalculate the mean.

runningMean()
    Return the running mean.

size()
    Return the size of the queue.

shape()
    Return the shape of images in the queue.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Index