Package 'spatialfil'

September 11, 2015

Type Package

Title Application of 2D Convolution Kernel Filters to Matrices or 3D Arrays

Version 0.15

Date 2015-09-08

Author Nicola Dinapoli, Roberto Gatta

Maintainer Nicola Dinapoli <nicola.dinapoli@rm.unicatt.it>

Description Filter matrices or (three dimensional) array data using different convolution kernels.

License GPL-2

LazyData TRUE

Depends R (>= 3.1.0), abind, fields, grDevices

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-09-11 17:10:49

R topics documented:

`applyFilter` *Function for applying convolution kernel to a matrix or array*

Description

This function applies the a convolution kernel based filter to a `matrix` or `array` object type.

Usage

```
applyFilter(x, kernel)
```

Arguments

x	An object of class matrix or array
kernel	A matrix containing the values chosen as convolution kernel

Details

The application of a convolution kernel over a 2D matrix dataset allows to apply functions as smoothing or edge detection. The aim of this function is to filter 2D matrices in order to help signal finding across (images-derived) data. It is also possible to filter 3D arrays considering them as slices of a series of images to be processed. Higher dimensions arrays are not allowed. The kernel parameter is a simple **square matrix** with an odd number of rows/columns, that can be pre-calculated by using the function convKernel. Not square matrices or matrices with even number of rows/columns will exit an error.

Value

An object with the same size of x containing data processed by convolution kernel

Examples

```
## Not run:
M <- array(runif(1000000), dim = c(100,100,100))
# smooth the array M
Mfil <- applyFilter(x = M, kernel = convKernel(sigma = 1.4, k = 'gaussian'))
image(M[,,50], col = grey(1:1000/1000))
image(Mfil[,,50], col = grey(1:1000/1000))

# now combining two filters in cascade
Mfil <- applyFilter(x = applyFilter(x = M, kernel = convKernel(k = 'sobel')),
                    kernel = convKernel(sigma = 1.4, k = 'gaussian'))
image(Mfil[,,50], col = grey(1:1000/1000))
## End(**Not run**)
```

---

convKernel *Function for creating convolution kernel for different filters*

Description

This function creates the convolution kernel for applying a filter to an array/matrix

Usage

```
convKernel(sigma = 1.4, k = c("gaussian", "LoG", "sharpen", "laplacian",
  "emboss", "sobel"))
```

Arguments

sigma	The numeric value of standard deviation for the Gaussian or LoG filter
k	character value:

- gaussian for Gaussian kernel
- LoG for Laplacian of Gaussian kernel
- sharpen for 3x3 convolution matrix for sharpening edges
- laplacian for a 3x3 convolution matrix that enhances the edges
- emboss for a 3x3 kernel that draws edges as embossed image
- sobel gives one of the two 3x3 matrices needed to apply the Sobel filter

Details

The convolution kernel is a matrix that is used by spacialfil function over a matrix, or array, for filtering the data. *Gaussian* kernel is calculated starting from the 2 dimension, isotropic, Gaussian distribution:

$$G(x) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Laplacian of Gaussian kernel applies a second derivative to enhance regions of rapid intensity changes:

$$LoG\,(x,y) = \frac{-1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$

the use of the underlying Gaussian kernel (so the name Laplacian of Gaussian or *LoG*) is needed to reduce the effect of high frequency noise that can affect the signal distribution. *Laplacian* is a *Sharpen* enhance the detail. *Emboss* kernel is a 3x3 convolution kernel that embosses the edges. (but also the noise) in original dataset. *Sobel* convolution kernel returns the possibility to detect edges in a more sofisticated way, the convKernel function returns only one of the two matrices needed to apply the filter. The second one is calculated by transposing the returned matrix in the other needed one.

Value

An object of class convKern with the matrix of convolution kernel whose size varies according the value of sigma (in case of gaussian or LoG option selected), and k being the convolution kernel type label

References

- gaussian kernel http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm
- LoG kernel: http://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm
- sharpen kernel: https://en.wikipedia.org/wiki/Kernel_(image_processing)
- laplacian kernel: https://en.wikipedia.org/wiki/Discrete_Laplace_operator
- emboss kernel: http://coding-experiments.blogspot.it/2010/07/convolution.html
- sobel kernel: https://en.wikipedia.org/wiki/Sobel_operator

Examples

```
## Not run:
# creates a convolution kernel with Gaussian function and sigma = 1.4
K <- convKernel(sigma = 1.4, k = 'gaussian')
plot(K)
## End(**Not run**)
```

Index