Package ‘imagine’

February 6, 2020

Type Package
Title IMAGing engINEs, Tools for Application of Image Filters to Data Matrices
Version 1.5.3
Date 2020-02-06
URL https://github.com/LuisLauM/imagine
BugReports https://github.com/LuisLauM/imagine/issues
Maintainer Wencheng Lau-Medrano <luis.laum@gmail.com>
Description Provides fast application of image filters to data matrices, using R and C++ algorithms.
License GPL (>= 2)
LazyData TRUE
Depends R (>= 3.1.0)
Imports Rcpp
LinkingTo Rcpp
RoxygenNote 7.0.2
Suggests knitr, rmarkdown
VignetteBuilder knitr
Encoding UTF-8
NeedsCompilation yes
Author Wencheng Lau-Medrano [aut, cre]
Repository CRAN
Date/Publication 2020-02-06 15:10:02 UTC

R topics documented:

  image-package .......................... 2
  contextualMF ............................ 2
  convolution2D ................................ 3
  meanFilter ................................ 4
  wbImage ................................ 5
**Index**

image-package | IMAGING.engINE, Tools for application of image filters to data matrices

**Description**

Provides fast application of image filters to data matrices, using R and C++ algorithms.

**Details**

This package uses C++ algorithms called 'engines'. More details are shown in the vignette.

**Author(s)**

Wencheng Lau-Medrano, <luis.laum@gmail.com>

---

**contextualMF** | Performs Contextual Median Filter

**Description**

This function performs the Contextual Median Filter proposed by Belkin & O’Reilly (2009), based on the pseudo-code written on the paper.

**Usage**

`contextualMF(X, times = 1, na = NA)`

**Arguments**

- `X` A numeric matrix object used for apply filters.
- `times` How many times do you want to apply the filter?
- `na` NA as default. But, if specified, it must be an integer value higher than the maximum of `X`.

**Details**

The users can change the number of recursive applications by using `times` argument (`times = 1` as default).

**Value**

`contextualMF` returns a matrix object with the same dimensions of `X`.
## convolution2D

**Make convolution calculations from numeric matrix**

### Description

This function takes a matrix object, and for each cell multiplies its neighborhood by the kernel. Finally, it returns for each cell the mean of the kernel-weighted sum.

### Usage

```r
convolution2D(X, kernel, times = 1, normalize = FALSE)
```

```r
convolutionQuantile(X, kernel, probs, times = 1, normalize = FALSE)
```

```r
convolutionMedian(X, kernel, times = 1)
```

### Arguments

- **X**: A numeric matrix object used for apply filters.
- **kernel**: A little matrix used as mask for each cell of X.
- **times**: How many times do you want to apply the filter?
- **normalize**: logical indicating if results will (or not) be normalized. See details.
- **probs**: numeric vector of probabilities with values in [0,1].
Details

Convolution is a mathematical operation which allows the multiplication of two arrays of numbers, in order to produce an array of numbers of the same dimensionality. Valid results (showed in output) will be only those with non-NA values, so NA holes on a matrix will expand in the order of the kernel size.

Normalization consists on divides the output by the sum(abs(as.numeric(kernel))) (disabled by default).

Value

convolution2D returns a matrix object with the same dimensions of X.

convolutionQuantile uses the kernel but, for each cell, it returns the position of quantile 'probs' (value between 0 and 1).

convolutionMedian is a wrapper of convolutionQuantile with probs = 0.5.

Examples

```r
# Generate example matrix
nRows <- 50
nCols <- 100

myMatrix <- matrix(runif(nRows*nCols, 0, 100), nrow = nRows, ncol = nCols)
kernel <- diag(3)

# Make convolution
myOutput1 <- convolution2D(myMatrix, kernel)
myOutput2 <- convolutionQuantile(myMatrix, kernel, probs = 0.7)

# Plot results
par(mfrow = c(2, 2))
image(myMatrix, zlim = c(0, 100))
image(myOutput1, zlim = c(0, 100))
image(myOutput2, zlim = c(0, 100))
```

meanFilter

Make a 2D filter calculations from numeric matrix

Description

This functions take a matrix object, and for each cell calculate mean, median or certain quantile about a squared neighborhood by matrix of dimension (radius + radius).

Usage

```r
meanFilter(X, radius, times = 1)

quantileFilter(X, radius, probs, times = 1, na = NA)

medianFilter(X, radius, times = 1, na = NA)
```
Arguments

- **X**: A numeric matrix object used for apply filters.
- **radius**: Size of squared kernel to apply median.
- **times**: How many times do you want to apply the filter?
- **probs**: numeric vector of probabilities with values in [0,1].
- **na**: NA as default. But, if specified, it must be an integer value higher than the maximum of X.

Details

Functions use C++ algorithms. More details are shown in the vignette.

Value

A matrix object with the same dimensions of X.

quantileFilter don’t use a kernel but, for each cell, it returns the position of quantile 'probs' (value between 0 and 1).

medianFilter is a wrapper of quantileFilter with probs = 0.5.

Examples

```r
# Generate example matrix
nRows <- 50
nCols <- 100

myMatrix <- matrix(runif(nRows*nCols, 0, 100), nrow = nRows, ncol = nCols)
radius <- 3

# Make convolution
myOutput1 <- meanFilter(X = myMatrix, radius = radius)
myOutput2 <- quantileFilter(X = myMatrix, radius = radius, probs = 0.1)
myOutput3 <- medianFilter(X = myMatrix, radius = radius)

# Plot results
par(mfrow = c(2, 2))
image(myMatrix, zlim = c(0, 100), title = "Original")
image(myOutput1, zlim = c(0, 100), title = "meanFilter")
image(myOutput2, zlim = c(0, 100), title = "quantileFilter")
image(myOutput3, zlim = c(0, 100), title = "medianFilter")
```

Description

matrix object containig numeric data to plot a image. The photo was taken by the author at 2016.
Usage
  wbImage

Format
  A matrix with dimensions 1280x720.
Index

contextualMF, 2
convolution2D, 3
convolutionMedian (convolution2D), 3
convolutionQuantile (convolution2D), 3

image-package, 2
imagine (image-package), 2
imagine-package (image-package), 2

meanFilter, 4
medianFilter (meanFilter), 4
quantileFilter (meanFilter), 4

wbImage, 5