Package 'wvtool'

November 8, 2016

Type Package

Title Image Tools for Automated Wood Identification

Version 1.0

Date 2016-11-07

Author Junji Sugiyama, Kayoko Kobayashi

Maintainer Kayoko Kobayashi <kayoko_kobayashi@rish.kyoto-u.ac.jp>

Description This tool, wood vision tool, is intended to facilitate preprocessing and analyzing 2-dimensional wood images toward automated recognition. The former includes some basics such as functions to RGB to grayscale, gray to binary, cropping, rotation(bilinear), median/mean/Gaussian filter, and Canny/Sobel edge detection. The latter includes gray level co-occurrence matrix (GLCM), Haralick parameters, local binary pattern (LBP), higher order local autocorrelation (HLAC), Fourier transform (radial and azimuthal integration), and Gabor filtering. The functions are intended to read data using 'readTIFF(x,info=T)' from 'tiff' package. The functions in this packages basically assumes the grayscale images as input data, thus the color images should be subjected to the function rgb2gray() before used for some other functions.

License GPL (>= 2)

NeedsCompilation no

Suggests tiff

Depends R (>= 2.10)

Repository CRAN

Date/Publication 2016-11-08 16:50:40

R topics documented:

`bin2dec` *Binary Number to Decimal number Conversion*

Description

A function returns decimal number from a sequential binary number. The function is internally used in lbp function.

Usage

```
bin2dec(x)
```

Arguments

`x`	A sequence of 0, 1 numbers

Value

A decimal number

See Also

dec2bin, lbp

camphora — *Image Sample Dataset*

Description

An optical micrographs of Cinnamomum camphora

Usage

```
data("camphora")
```

Format

The format is: num [1:486, 1:518] 0.275 0.337 0.765 0.937 0.933 ... - attr(*, "bits.per.sample")= int 8 - attr(*, "samples.per.pixel")= int 1 - attr(*, "sample.format")= chr "uint" - attr(*, "planar.config")= chr "contiguous" - attr(*, "compression")= chr "none" - attr(*, "x.resolution")= num 32 - attr(*, "y.resolution")= num 32 - attr(*, "resolution.unit")= chr "inch" - attr(*, "orientation")= chr "top.left" - attr(*, "color.space")= chr "black is zero"

Source

Kyoto University Xylarium Database

References

http://database.rish.kyoto-u.ac.jp

Examples

```
data(camphora)
## maybe str(camphora)
```

car2pol — *Polar Transformer -Cartesian to Polar Coordinates-*

Description

The function converts images to polar coordintates. The polar transformation is useful for unwarping images which have a generally round object. From power specturm for example, one may generate radial integration profile or azimuthal intensity distribution. Default is "bilinear" interpolation.

Usage

```
car2pol(x, method="bilinear")
```

Arguments

`x`	A raster image or a matrix.
`method`	"NN" Nearest neighbour method, "bilinear" Bilinear interpolation.

Value

A matrix in polar coordinate system of the requested image

`pol.img`	Radial distance corresponds to the shorter side of requested image, and polar angle covers 0 to 360 degrees.

See Also

integ.profile

Examples

```
data(camphora)
par(mfrow=c(1,2))
image(rot90c(camphora), col=gray(c(0:255)/255), main="camphora, original",
    useRaster=TRUE, asp=1, axes=FALSE)
img <- car2pol(camphora, method="bilinear")
image(rot90c(img), col=gray(c(0:255)/255), main="camphora, polar (bilinear)",
xlab="radial distance(pixel)",ylab="angle(deg)", useRaster=TRUE, asp=1, axes=FALSE)
```

`cc.label` *Connected Component Labelling.*

Description

A function labels the connected components in a binary image. For example, it can be used for statistical analysis of tracheids (see examples).

Usage

```
cc.label(x, connect=8, inv=FALSE, img.show=FALSE,text.size=0.3)
```

Arguments

`x`	A binary image (A matrix with 0 and 1)
`connect`	8-connectivity or 4-connectivity. Default is 8-connectivity.
`inv`	inverse the binary image x before labelling. Labelling the connected area with 0 when this is TRUE.
`img.show`	If this is TRUE, the image with labelling numbers are shown.
`text.size`	the size of labelling numbers used when img.show=TRUE.

Details

Labelling the connected components with pixels equal to 1 (white) in a binary image (If pixels equal to 0 (black) should be labelled, select inv=TRUE). The function returns the labelled image and the statistical data of the labelled components.

Value

a list with 2 components (a matrix and a dataframe)

`image`	A matrix with labels
`summary`	A dataframe summarizing the labelled components with 8 following variables.
`summary$label`	labelling numbers, area: area of each component
`summary$aveX, summary$aveY`	center position of each component
`summary$dX, summary$dY`	width and height of each component
`summary$edge`	If the component is on the edge of the image, this value is 1, otherwise 0.

Examples

```
## Not run:
data(cryptomeria)
img <- rgb2gray(cryptomeria)
img.c <- crop(img,300,300)
img.bin <- gray2bin(img.c, auto=FALSE, th=180)
par(mfrow=c(2,2))
test <- cc.label(img.bin, connect=8, img.show=TRUE)
hist(test$summary$area,main="histogram of area")
hist(test$summary$dX,main="histogram of dX")
hist(test$summary$dY,main="histogram of dY")

## End(Not run)
```

`crop` *Image cropping*

Description

image cropping from the center.

Usage

```
crop(x, width=300, height=300, shift=c(0,0))
```

Arguments

`x`	A raster or a matrix
`width`	width for cropping
`height`	height for cropping
`shift`	shift of the cropped position from the center

Value

A raster or a matrix

Examples

```
data(camphora)
par(mfrow=c(2,2))
image(rot90c(camphora),col=gray(c(0:255)/255), main="original", useRaster=TRUE, axes=FALSE, asp=1)
image(rot90c(crop(camphora,200,100)),col=gray(c(0:255)/255),
main="cropped from the center", useRaster=TRUE, axes=FALSE, asp=0.5)
image(rot90c(crop(camphora,200,200)),col=gray(c(0:255)/255),
main="cropped from the center", useRaster=TRUE, axes=FALSE, asp=1)
image(rot90c(crop(camphora,200,200,shift=c(50,50))),col=gray(c(0:255)/255),
main="cropped from shifted position c(50,50)", useRaster=TRUE, axes=FALSE, asp=1)
```

`cryptomeria` *Image Sample Dataset*

Description

An optical micrographs of Cryptomeria japonica

Usage

```
data("cryptomeria")
```

Format

num [1:1079, 1:1000, 1:4] 0.886 0.89 0.863 0.639 0.424 ... - attr(*, "bits.per.sample")= int 8 - attr(*, "samples.per.pixel")= int 4 - attr(*, "sample.format")= chr "uint" - attr(*, "planar.config")= chr "contiguous" - attr(*, "compression")= chr "none" - attr(*, "x.resolution")= num 72 - attr(*, "y.resolution")= num 72 - attr(*, "resolution.unit")= chr "inch" - attr(*, "orientation")= chr "top.left" - attr(*, "artist")= chr "DP" - attr(*, "date.time")= chr "2016:07:29 11:38:28" - attr(*, "color.space")= chr "RGB"

Source

Kyoto University Xylarium Database

References

http://database.rish.kyoto-u.ac.jp

Examples

```
data(cryptomeria)
## maybe str(cryptomeria)
```

---

dec2bin *Decimal to Binary Convertion*

Description

A function returns binary number from decimal number. The function is internally used in lbp function.

Usage

```
dec2bin(x, digit=8)
```

Arguments

x	A decimal integer.
digit	A length of binary sequence.

Value

A binary number in array.

See Also

bin2dec, lbp

---

edge.detect *Canny and Sobel Edge detector.*

Description

A function detects edges in images by Canny or Sobel operator. Sobel provides approximate intensity of gradients for each pixels, while Canny provides a binary image with thin edges.

Usage

```
edge.detect(x, thresh1=1, thresh2=15, noise="gaussian", noise.s=3, method="Canny")
```

Arguments

x	A raster image or a matrix
thresh1	low threshold for edge tracking by hysteresis (0-100). Only used for "Canny" edge detector.
thresh2	high threshold for edge tracking by hysteresis (0-100). Only used for "Canny" edge detector.
noise	a method for noise reduction. "gaussian", "median", and "mean" filters are available. Default is "gaussian".
noise.s	filter size for noise reduction (3 or 5). Default is 3.
method	"Canny" and "Sobel" can be selected. Default is "Canny".

Details

Canny edge detector has four steps. 1. noise reduction/ 2. finding the gradient in images by Sobel operator/ 3. Non-maximum suppression/ 4. Hysteresis threshold. When the method "Sobel" is selected, only step 1 and 2 will be done.

Value

A raster or a matrix

See Also

noise.filter

Examples

```
## Not run:
data(camphora)
data(cryptomeria)
cryptomeria <- rgb2gray(cryptomeria)
img.c1 <- crop(camphora,200,200)
img.c2 <- crop(cryptomeria,300,300)
par(mfrow=c(2,2))
image(rot90c(edge.detect(img.c1,thresh1=1, thresh2=15, noise="gaussian", noise.s=3,
 method="Canny")),col=gray(c(0:255)/255), main="Canny", useRaster=TRUE, axes=FALSE, asp=1)
image(rot90c(edge.detect(img.c1,thresh1=1, thresh2=15, noise="gaussian", noise.s=3,
 method="Sobel")),col=gray(c(0:255)/255), main="Sobel", useRaster=TRUE, axes=FALSE, asp=1)
image(rot90c(edge.detect(img.c2,thresh1=1, thresh2=15, noise="gaussian", noise.s=3,
 method="Canny")),col=gray(c(0:255)/255), main="Canny", useRaster=TRUE, axes=FALSE, asp=1)
image(rot90c(edge.detect(img.c2,thresh1=1, thresh2=15, noise="gaussian", noise.s=3,
 method="Sobel")),col=gray(c(0:255)/255), main="Sobel", useRaster=TRUE, axes=FALSE, asp=1)

## End(Not run)
```

---

gabor.filter   *Two Dimensional Gabor Filtering in Frequency Domain*

Description

The function provides two dimensional Gabor function proposed by Daugman to model the spatial summation properties in the visual cortex. It returns Gabor filter in real and reciprocal space, and filtered image.

Usage

```
gabor.filter(x, lamda=5, theta=45, bw=1.5, phi=0, asp=1, disp=FALSE)
```

Arguments

`x`	A raster or matrix to be filtered
`lamda`	Wavelength of the cosine part of Gabor filter kernel in pixel. Real number greater than 2 can be used. However, lamda=2 should not be used with phase offset (phi) = -90 or 90.
`theta`	The orientation of parallel stips of Gabor function in degree
`bw`	Half responce spatial frequency bandwidth of a Gabor filter. This relates to the ratio sigma/lamda, where sigma is the standard deviation of Gaussian factor of Gabor function.
`phi`	Phase offset of the cosine part of Gabor filter kernel in degree
`asp`	Elliptcity of the Gabor function. asp=1 means circular. For asp<1 it gives elongated parallel strip
`disp`	If this operator is TRUE, original image, gabor filter in real space domain, that in frequency domain, and filtered image will be generated

Value

The function provides following four outputs.

`kernel`	151x151 gabor filter kernel
`mask`	A mask with kernel in the center in a space domain
`freq_mask`	Real part of Fourier transform of the mask in spatial frequency domain
`filtered_image`	Inversed Fourier transform of FFT(img)*FFT(mask)

References

J. G. Daugman, Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two d-dimensional visual cortical filters, J. Opt. Sppc. Am A, 2(7), 1160-1169, 1985

Examples

```
data(cryptomeria)
img <- rgb2gray(cryptomeria)
img <- crop(img,300,300)
# simple example
test <- gabor.filter(x=img, lamda=8, theta=60, bw=1.5, phi=0, asp=0.3, disp=TRUE)
## Not run:
# azimuthal intensity distribution with respect to the orientation of Gabor function
par(mfrow=c(2,1))
Integ <- array()
filt.img <- matrix(0, nrow(img), ncol(img))
for ( i in 1:60) {
out <- gabor.filter(x=img, lamda=8, theta=3*i, bw=1.5, phi=0, asp=0.3)
filt.img <- out$filtered_img + filt.img
Integ[i] <- sum(out$filtered_img*out$filtered_img)
}
image(rot90c(filt.img),col=gray(c(0:255)/255),asp=1,axes=FALSE, useRaster=TRUE)
x <- 1:60
plot(3*x,Integ, ty="l", ylab="integrated intensity (a.u.)", xlab="azimuthal angle (deg)")

## End(Not run)
```

---

glcm — *Gray Level Co-occurence Matrix (glcm)*

Description

This function supports calculating gray level co-occurence matrices from a grayscale image (< 8 bit) with requested gray level. The gray level of the source image is read from the attributes data from input TIFF file.

Usage

```
glcm(x, t.level=4, d=1)
```

Arguments

`x`	A gray scale image or matrix. "x" assumes an output from readTIFF(filename, as.is=T, info=T)
`t.level`	A target grey level for GLCM calculation in bite. The grayscale is truncated linearly.
`d`	Displacement between adjacent i, j points in pixel.

Details

The data in matrix is either inspected as images or subsequently used to calculate Haralick texture features, originally published 15 features(Haralick et al., 1973) and two additionals (Albregsten, 1995).

Value

The gray level cooccurence matrices of 4 directions (theta) and their average, gray level, and displacement vector are listed.

`glcm`	GLCM at theta = "0", " 45", "90", "135" degree and "average"
`level`	numbr of gray level
`d`	length of displacement vector

References

R.M. Haralick, K. Shangmugam, Its'hak Dinstein (1973) Textural Features for Image Classification, IEEE Transactions on Systems, Man, and Cybernetics, SMC-3(6), 610-621.

Albregtsen F (1995) Statistical texture measures computed from gray level cooccurrence matrices. In: Technical Note, Department of Informatics, University of Oslo, Norway

K. Kobayashi, M. Akada, T. Torigoe, S. Imazu, J. Sugiyama (2015) Automated recognition of wood used in traditional Japanese sculptures by texture analysis of their low-resolution computerd tomography data. J. Wood Sci., 61, 630-640

See Also

gray2bin, rgb2gray, haralick,

Examples

```
data(camphora)
img <- camphora
par(mfrow=c(1,2))
lev <- 4
theta <- c(1,3)  # "th_0","th_90"
theta_c <-c("th_0","th_90")
dist <- 1
for (i in 1:2) {
tst <- glcm(img,lev,dist)
title <- paste(lev, "bit", " glcm ", theta_c[i], " d=", dist, sep="")
persp(tst$glcm[[i]], theta=30, phi=30,main=title, asp=1,
xlab="i", ylab="j", zlab="probability")
}
```

`gray2bin` *Conversion from Grayscale to Binary Image*

Description

A function provides automatic clustering-based thresholding proposed by Ohtsu, and a gray scale image is converted to binary image. Initial histogram and discriminant level of binning are displayed by his=TRUE, dis=TRUE options. A threshold value can be also set manually.

Usage

```
gray2bin(x, auto=TRUE, th=200, his=FALSE, dis=FALSE)
```

Arguments

`x`	A raster image or a matrix
`auto`	set threshold automatically (Ohtsu method) or manually
`th`	a threshold value used when auto=FALSE
`his`	A histogram of initial gray scale image
`dis`	A plot of variation between classes divided by variation within classes

Value

A requested binary image. Black is zero.

References

N. Otsu (1979) A threshold selection method from gray-level histograms, IEEE Trans. Sys., Man., Cyber., 9(1), 62-66.

See Also

rgb2gray

Examples

```
data(camphora)
par(mfrow=c(2,3))
image(rot90c(camphora), col= gray((0:255)/255), main="camphora", asp=1, useRaster=TRUE, axes=FALSE)
out <- gray2bin(camphora, his=TRUE, dis=TRUE)
image(rot90c(out), col= gray((0:255)/255), main="binary image, auto", asp=1,
useRaster=TRUE, axes=FALSE)
image(rot90c(gray2bin(camphora,auto=FALSE,th=100)), col= gray((0:255)/255), main="binary image,
thresh=100", asp=1, useRaster=TRUE, axes=FALSE)
image(rot90c(gray2bin(camphora,auto=FALSE,th=180)), col= gray((0:255)/255), main="binary image,
thresh=180", asp=1, useRaster=TRUE, axes=FALSE)
```

`haralick` *Haralick Texture Features Calculated from GLCM*

Description

A function returns 15 Halarick features for 4 directions, their average and range.

Usage

```
haralick(x)
```

Arguments

`x`	output of glcm() function from a TIFF data

Details

15 outputs are #1 Angular Second Moment / Homogeniety "asm" #2 Contrast "con" #3 inverse Difference Moment "idm" #4 Entropy "ent" #5 Correlation "cor" #6 Variance in Haralick 1973 "var" #7 Sum Average "sav" #8 Sum Entropy "sen" #9 Difference Entropy "den" #10 Difference Variance "dva" #11 Sum Variance "sva" #12 Information Measures of Correlation "f12" #13 Information Measures of Correlation "f13" #14 Cluster Shade "sha" #15 Cluster prominence "pro", respectively

Value

A matrix of angles and features

References

R.M. Haralick, K. Shangmugam, Its'hak Dinstein (1973) Textural Features for Image Classification, IEEE Transactions on Systems, Man, and Cybernetics, SMC-3(6), 610-621.

Albregtsen F (1995) Statistical texture measures computed from gray level cooccurrence matrices. In: Technical Note, Department of Informatics, University of Oslo, Norway

K. Kobayashi, M. Akada, T. Torigoe, S. Imazu, J. Sugiyama (2015) Automated recognition of wood used in traditional Japanese sculptures by texture analysis of their low-resolution computerd comography data. J. Wood Sci., 61, 630-640.

See Also

glcm

Examples

```
data(camphora)
haralick(glcm(camphora,6,1))
```

`hlac` *Higher Order Local Autocorrelation (HLAC)*

Description

Feature extraction for practical vision system, whose features are shift-invariant and additive. The function gives zero to the eighth order cases, represented by 223 mask patterns of 3 x 3 within a 2r+1 x 2r+1 (r >=1) displacement region.

Usage

```
hlac(x, r=1, disp=FALSE)
```

Arguments

x	A binary or gray image or matrix
r	Displacement vector r for 3 x 3 mask pattern
disp	If TRUE, function saves 223 filtered images in one matrix.

Details

The feature parameter should be a list. The function returns 1, 4, 20, 45, 62, 54, 28, 8,1 features and corresponding filtered images if disp is TRUE.

Value

HLAC features or the corresponding image with requested HLAC measures.

features	Numerical output of 0 to 8th order masks
mat	A large matrix of 223 images expanded in a row

References

N.Otsu, T. Kurita (1988) A New Scheme for Practical Flexible and Intelligent Vision Systems, In: Proc. Machine Vision Application(MVA), 431-435.

See Also

rgb2gray, gray2bin, glcm, lbp

Examples

```
# features plot and the corresponding image presentation
data(camphora)
tmp <- hlac(gray2bin(camphora), 2, disp=TRUE)
par(mfrow=c(2,2))
plot(unlist(tmp$features), main="HLAC histogram")
image(rot90c(matrix(tmp$mat[2,],tmp$row,tmp$col)),
col  = gray((255:0)/255), main="2", useRaster=TRUE, asp=1, axes=FALSE)
image(rot90c(matrix(tmp$mat[23,],tmp$row,tmp$col)),
col  = gray((255:0)/255), main="23", useRaster=TRUE, asp=1, axes=FALSE)
image(rot90c(matrix(tmp$mat[156,],tmp$row,tmp$col)),
col  = gray((255:0)/255), main="156", useRaster=TRUE, asp=1, axes=FALSE)
```

integ.profile *Simple Integration for Making Profile*

Description

A function returns integrated line profile. It crops rectangular area from a requested size and project and integrate pixel values either to horizontal or vertical axis. When used with a matrix in polar coordinate (car2pol) calculated from power spectrum (power.spec) of an image, the function provides radial integration or azimuthal integration that are useful for diffraction analysis.

Usage

```
integ.profile(x, axis="H", h=c(20, 50), v=c(30, 120), disp=FALSE)
```

Arguments

`x`	A raster image or a matrix
`axis`	Axis to project. H : Projection to horizontal axis or radial distance (in polar coordinate). V : Projection to vertical axis or azimuthal angle (in polar coordinate).
`h`	c(h1,h2) : A horizontal or radial (in polar coordinate) range for integration.
`v`	c(v1,v2) : A vertical or azimuthal (in polar coordinate) range for integration.
`disp`	Plot calculated profile. Default is FALSE.

Details

The row and column corresponds to horizontal and vertical axes, respectively.

Value

An array of requested line profile

See Also

swap.quad, car2pol, fft, Mod

Examples

```
data("camphora")
img <- camphora
par(mfrow=c(2,2))
image(rot90c(img),col=gray(c(0:255)/255), useRaster=TRUE, main="camphora",asp=1, axes=FALSE)
integ.profile(img, axis="H", h=c(1,nrow(img)) , v=c(1,ncol(img)), disp=TRUE)
integ.profile(img, axis="V", h=c(1,nrow(img)) , v=c(1,ncol(img)), disp=TRUE)
ps <- log(swap.quad(Mod(fft(img))))
pol <- car2pol(ps)
image(rot90c(ps),col=gray(c(0:255)/255), useRaster=TRUE,main="power spectrum",asp=1, axes=FALSE)
```

```
image(rot90c(pol),col=gray(c(0:255)/255), useRaster=TRUE,main="polar map",asp=1)
integ.profile(pol, axis="H", h=c(10,200) , v=c(0,90), disp=TRUE)
integ.profile(pol, axis="V", h=c(70,100) , v=c(0,360), disp=TRUE)
```

`lbnum` *Counts 0-1 or 1-0 in a Binary Sequence*

Description

A function returns how many 0-1 or 1-0 transitions in a binary sequence. For example, 00010000 is 2 transition and 01010100 is 6 transitions. It is internally used in the lbp function.

Usage

```
lbnum(seq)
```

Arguments

`seq`	A sequential 0, 1 array

See Also

lbp

`lbp` *Local Binary Patterns (lbp)*

Description

Calculate local binary patterns from a grayscale image

Usage

```
lbp(x, r=1)
```

Arguments

`x`	A raster image or a matrix
`r`	displacement vector in 8 direction. r=1 means c(-1, 0, -1,1, 0,1,1,1,1,0,1,-1,0,-1,-1,-1) r=2 means c(-2, 1, -1,2, 1,2,2,1,2,-1,1,-2,-1,-2,-2,-1)

Details

The LBP operator was originally designed for texture description. The operator assigns a label to every pixel of an image by thresholding the 3x3-neighborhood of each pixel with the center pixel value and considering the result as a binary number (gives 0 if each pixel is smaller than the center, otherwise 1). Then, the histogram of the labels can be used as a texture descriptor. The circular (8,r=1), and (8,r=2) neighborhoods are considered. The function assumes 8-bit grayscale image as an input.

Value

`lbp.u2`	a matrix (image) returned from requested LBP u2 operation
`lbp.ori`	a matrix (image) returned from requested LBP operation

Note

A local binary pattern is called uniform if the binary pattern contains at most two 0-1 or 1-0 transitions. In calculation, the histogram has 58 separate bins for uniform patterns, and all other non-uniform patterns are assigned to one single bin. Thus, the length of the features reduces from 256 to 59. This function returned both the lbp patterns before (lbp.ori) and after (lbp.u2) considering the uniform patterns.

References

T. Ojala, M. Pietikainen, and D. Harwood (1994), Performance evaluation of texture measures with classification based on Kullback discrimination of distributions, Proceedings of the 12th IAPR International Conference on Pattern Recognition(ICPR 1994), vol.1, pp. 582-585.

T. Ojala, M. Pietikainen, and D. Harwood (1996) A Comparative Study of Texture Measures with Classification Based on Feature Distributions, Pattern Recognition, vol. 29, no. 1, 51-59.

See Also

rgb2bin, hlac

Examples

```
## Not run:
data(camphora)
par(mfrow=c(2,2))
r1 <- lbp(camphora,1)
image(rot90c(r1$lbp.u2),col = gray((0:58)/58), main="lbp.u2 (r=1, 8 points)", useRaster=TRUE,
asp=1, axes=FALSE)
image(rot90c(r1$lbp.ori),col = gray((0:255)/255), main="lbp.ori (r=1, 8 points)", useRaster=TRUE,
asp=1, axes=FALSE)
hist(r1$lbp.u2,breaks=59, main="Histogram of lbp.u2")
hist(r1$lbp.ori,breaks=256, main="Histogram of lbp.ori")

## End(Not run)
```

`noise.filter` *Median, Mean and Gaussian Filter*

Description

A funtion provides three kinds of noise reduction on an image, "median", "mean", and "gaussian". A typical pre-processing step to improve the results of later processing for example, glcm-haralick analysis.

Usage

```
noise.filter(x, n=3, method="median")
```

Arguments

`x`	A raster image or a matrix
`n`	filter size is given by n x n. Default is 3 x 3. Number has to be an odd number. For gaussian filter, only 3 or 5 is available.
`method`	"median", "mean", and "gaussian" can be selected. Default is "median".

Value

A raster or a matrix

References

T.S. Huang, G.J. Yang, G.Y. Tang (1979) A fast two-dimensional median filtering algorithm, IEEE transactions, Acoustics, Speech and Signal Processing, 27, 13-18.

See Also

glcm

Examples

```
data(camphora)
camphora <- crop(camphora,200,200)
par(mfrow=c(2,2))
image(rot90c(noise.filter(camphora,3,"median")),col=gray(c(0:255)/255),
main="median", useRaster=TRUE, axes=FALSE, asp=1)
image(rot90c(noise.filter(camphora,3,"mean")),col=gray(c(0:255)/255),
main="mean", useRaster=TRUE, axes=FALSE, asp=1)
image(rot90c(noise.filter(camphora,3,"gaussian")),col=gray(c(0:255)/255),
main="gaussian", useRaster=TRUE, axes=FALSE, asp=1)
```

rgb2gray *Convert RGB image to Grayscale*

Description

A function returns grayscale image with coefficients = c(0.3, 0.59, 0.11).

Usage

```
rgb2gray(x, coefs=c(0.30, 0.59, 0.11))
```

Arguments

`x`	A raster image or a matrix
`coefs`	R, G, B weights. Default are coefs=c(0.30, 0.59, 0.11)

Value

A grayscale image

See Also

gray2bin

rot90c *Transpose Image 90 Degrees Clockwisely*

Description

Maybe useful to visualize images read by tiff::readTIFF using graphic::image

Usage

```
rot90c(x)
```

Arguments

`x`	A raster or a matrix

`rotate.matrix` *Image Rotation by Bilinear Interpolation*

Description

Three methods to execute rotation by 1) assuming values to destination, 2) obtaining values from the source image by inverse rotation with "nearest neighbor (NN)", 3) previous procedure together with "bilinear interpolation". The default is a rotation with "bilinear Interpolation".

Usage

```
rotate.matrix(x, angle, method="bilinear")
```

Arguments

`x`	A raster image or a matrix
`angle`	Plus(>0) value to request clockwise rotation, while minus for anticlockwise rotation.
`method`	"simple" assumes values to destination', "NN" obtains values from the source image by inverse rotation with "nearest neighbor", and "bilinear" performs the same but with "bilinear interpolation" of the source image. value to request clockwise rotation.

Details

Assuming 8-bit grayscale image as an input.

Value

A matrix after rotation

See Also

rgb2gray

Examples

```
data(camphora)
par(mfrow=c(2,2))
r1 <- rotate.matrix(camphora,15, method="simple")
image(rot90c(r1),asp=1,col=grey(c(0:255)/255), main= "simple",
useRaster=TRUE, axes=FALSE)
r2 <- rotate.matrix(camphora,25, method="NN")
image(rot90c(r2),asp=1,col=grey(c(0:255)/255), main="nearest neighbour", useRaster=TRUE, axes=FALSE)
r3 <- rotate.matrix(camphora,35, method="bilinear")
image(rot90c(r3),asp=1,col=grey(c(0:255)/255),main="bilinear interpolation",
useRaster=TRUE, axes=FALSE)
```

swap.quad *Swapping Quadrants*

Description

A function maybe useful to generates power spectrum from fft output.

Usage

```
swap.quad(x, disp=FALSE, reverse=FALSE)
```

Arguments

`x`	output of Mod(fft(imagefile))
`disp`	TRUE requests to draw power spectrum
`reverse`	TRUE should be used when power spectrum of N x M, where one of them is odd number.

Value

a matrix of power spectrum

See Also

fft, Mod

Examples

```
data(camphora)
data(cryptomeria)
img1 <-camphora
img2 <- rgb2gray(cryptomeria)
par(mfrow=c(2,2))

image(rot90c(img1),col=gray(c(0:255)/255), main="Camphora", asp=1,
useRaster=TRUE, axes=FALSE)
o.fft <-Mod(fft(img1))
ps <- swap.quad(o.fft)
image(rot90c(log(ps)),col=gray(c(0:255)/255), main="power spectrum", asp=1,
useRaster=TRUE, axes=FALSE)

image(rot90c(img2),col=gray(c(0:255)/255), main="Cryptomeria", asp=1,
useRaster=TRUE, axes=FALSE)
image(rot90c(log(swap.quad(Mod(fft(img2))))),col=gray(c(0:255)/255),
main="power spectrum", asp=1, useRaster=TRUE, axes=FALSE)
```

Index