

Package ‘rimage’

June 5, 2009

Version 0.5-8

Date 2005-1-12

Title Image Processing Module for R

Author Nikon Systems Inc.

Maintainer ORPHANED

Depends R (>= 1.6)

Description This package provides functions for image processing, including sobel filter, rank filters, fft, histogram equalization, and reading JPEG file. This package requires fftw-2 <<http://www.fftw.org/>> and libjpeg <<http://www.ijg.org/>>. This version doesn't require pixmap package, which the older version of rimage (private only) required. This package can be used on Unixes / MacOS X / Windows.

License BSD

Repository CRAN

Date/Publication 2009-06-05 07:20:02

R topics documented:

cat	2
clipping	3
equalize	3
fftImg	4
fftw	5
highpass	5
imagematrix	6
imageType	7
laplacian	8
logo	9
lowpass	9
meanImg	10

normalize	10
plot.imagematrix	11
print.imagematrix	11
Rank Filters	12
read.jpeg	13
rgb2grey	13
sobel	14
sobel.h	14
sobel.v	15
thresholding	16
Index	17

cat	<i>JPEG picture of a cat</i>
-----	------------------------------

Description

JPEG file of a cat.

Usage

```
cat
```

Format

JPEG

Source

Original photograph taken by a member of rimage developer.

See Also

[read.jpeg](#)

Examples

```
data(cat)
# Same as: x <- read.jpeg(system.file("data", "cat.jpg", package="rimage"))
plot(cat)
```

`clipping`*Clipping image*

Description

This function returns the image which restricts pixel value from the specified lowest value to the specified highest value in the original image. This means that the pixels which have lower value than the given lowest (default: 0) are replaced to the lowest and the pixels have greater value than the given highest (default: 1) are replaced to the highest.

Usage

```
clipping(img, low=0, high=1)
```

Arguments

<code>img</code>	target image
<code>low</code>	lowest value
<code>high</code>	highest value

Value

Data of the same mode as 'img'

Examples

```
data(logo)
op <- par(mfrow=c(2,2))
plot(logo, main="Source Image")

# the appearance of next one doesn't change because of normalization.
plot(normalize(2*logo), main="Doubled pixel value with normalization")

# the next one is saturated as expected
plot(clipping(2*logo), main="Doubled pixel value with clipping")
```

`equalize`*Make image having equalized histogram*

Description

This function make image having equalized histogram of original image.

Usage

```
equalize(img)
```

Arguments

img target image

Value

Data of the same mode as 'img', in which minimum value is 0 and maximum value is 1.

Examples

```
op <- par(mfrow=c(2,2))
data(logo)

plot(logo, main="Original Logo")
plot(equalize(logo), main="Equalized Logo")

catpic <- rgb2grey(read.jpeg(system.file("data", "cat.jpg", package="rimage")))
plot(catpic, main="Cat Image in Greyscale")
plot(equalize(catpic), main="Equalized Cat Image")

par(op)
```

fftImg

Compute FFT image

Description

This function computes the power spectrum of a given image by FFT.

Usage

```
fftImg(img)
```

Arguments

img target image

Value

an imagematrix

References

FFTW (Fastest Fourier Transform in the West) <http://www.fftw.org>

See Also

[fftw,imagematrix](#)

Examples

```
data (logo)
plot (normalize (fftImg (logo)))
```

`fftw`*Apply FFT to 2-Dimensional Data*

Description

This function applies FFT to 2-dimensional data (i.e. image) using fftw library.

Usage

```
fftw (img, dir = -1, debug=FALSE)
```

Arguments

<code>img</code>	target image
<code>dir</code>	set -1 for normal transformation and 1 for inverse transformation
<code>debug</code>	set TRUE if you want to output debug message

Value

a matrix of complex number

References

FFTW (Fastest Fourier Transform in the West) <http://www.fftw.org>

See Also

[fftw](#)

`highpass`*High pass filter for image*

Description

Computes a high-pass filtered image with dimensions of the given input image. the hp passing distance is given via radius.

Usage

```
highpass (img, radius)
```

Arguments

<code>img</code>	pixmap or image
<code>radius</code>	gives the blocking radius

Value

an `imagematrix`

See Also

[lowpass](#)

Examples

```
data (logo)
plot (normalize (highpass (logo)))
```

`imagematrix`

Generate an `imagematrix`, i.e. primary data structure of `rimage`

Description

This function makes an `imagematrix` object from a matrix. This data structure is primary data structure to represent image in `rimage` package.

Usage

```
imagematrix(mat, type=NULL,
            ncol=dim(mat)[1], nrow=dim(mat)[2], noclipping=FALSE)
```

Arguments

<code>mat</code>	array, matrix or vector
<code>type</code>	"rgb" or "grey"
<code>ncol</code>	width of image
<code>nrow</code>	height of image
<code>noclipping</code>	TRUE if you disable automatic clipping. See details.

Details

For grey scale image, matrix should be given in the form of 2 dimensional matrix. First dimension is row, and second dimension is column.

For rgb image, matrix should be given in the form of 3 dimensional array (row, column, channel). `mat[,1]`, `mat[,2]`, `mat[,3]` are red plane, green plane and blue plane, respectively.

You can omit 'type' specification if you give a proper array or matrix. Also, if you give a rgb image matrix and specify "grey" as type, the rgb image matrix is automatically converted to a grey scale image.

This function automatically clips the pixel values which are less than 0 or greater than 1. If you want to disable this behavior, give 'noclipping=TRUE'.

The major difference between `imagematrix` and `pixmap` is representation method. `pixmap` (>0.3) uses OOP class. On the other hand, `rimage` uses traditional S class. The advantage of traditional S class in representing image is that one can deal with the data structure as an ordinary matrix.

The minor difference between `imagematrix` and `pixmap` is automatic data conversion behavior. `pixmap` normalizes a given matrix automatically if any element of the matrix is out of range between 0 and 1. On the other hand, `imagematrix` clips the matrix, which means that the pixels which have lower value than 0 are replaced to 0 and the pixels have greater value than 1 are replaced to 1.

Value

return an `imagematrix` object

See Also

`plot.imagematrix`, `print.imagematrix`, `clipping`, `normalize`

Examples

```
p <- q <- seq(-1, 1, length=20)
r <- 1 - outer(p^2, q^2, "+") / 2
plot(imagematrix(r))
```

imageType

Get information on color type of imagematrix

Description

This function returns color type ("rgb" or "grey") of a given `imagematrix`.

Usage

```
imageType(x)
```

Arguments

x target image

Value

"rgb" or "grey"

Examples

```
x <- read.jpeg(system.file("data", "cat.jpg", package="rimage"))
cat("Image Type", imageType(x))
```

```
x.grey <- rgb2grey(x)
cat("Image Type", imageType(x.grey))
```

laplacian

Laplacian of image

Description

Calculate 2nd derivative of image for edge extraction

Usage

```
laplacian(img)
```

Arguments

img target image

Value

an object of pixmap class whose the size is as same as target

Examples

```
data(logo)
plot(normalize(laplacian(logo)))
```

`logo`*R logo imagematrix*

Description

The imagematrix object of R logo of the size 101x77.

Usage

```
data (logo)
```

Format

imagematrix

Examples

```
data (logo)
plot (logo)
```

`lowpass`*Low Pass Filter for Image*

Description

Computes a low-pass filtered image with dimensions of the given input image. the lp passing distance is given via radius.

Usage

```
lowpass (img, radius)
```

Arguments

<code>img</code>	pixmap or matrix
<code>radius</code>	gives the pass radius

Value

an imagematrix

See Also

[highpass](#)

Examples

```
data (logo)
plot (normalize (lowpass (logo) ) )
```

meanImg

Mean filter

Description

This function applies mean filter among 9 neighbors to a given image.

Usage

```
meanImg (img)
```

Arguments

img target image

Value

object of `imagematrix` class whose the size is as same as target

Examples

```
data (logo)
plot (meanImg (logo) )
```

normalize

Normalization for vector and matrix

Description

This function normalizes image so that the minimum value is 0 and the maximum value is 1.

Usage

```
normalize (img)
```

Arguments

img target image

Value

Data of the same mode as 'img', in which minimum value is 0 and maximum value is 1.

Examples

```
data(logo)
plot(normalize(logo))
```

plot.imagematrix *Plotting an imagematrix object*

Description

This function outputs an imagematrix object as an image.

Usage

```
## S3 method for class 'imagematrix':
plot(x, ...)
```

Arguments

x	target image
...	plotting options

See Also

[imagematrix](#)

Examples

```
op <- par(mfrow=c(1,2))

data(logo)
plot(logo, main="plot(logo)")
plot(logo^2, main="plot(logo^2)")

par(op)
```

print.imagematrix *Print information on a given imagematrix object*

Description

This function outputs information on a given imagematrix object.

Usage

```
## S3 method for class 'imagematrix':
print(x, ...)
```

Arguments

`x` target image
`...` ignored (dummy)

See Also

`imagematrix`

Examples

```
data(logo)
print(logo)
```

Rank Filters

Rank filters (minImg and maxImg)

Description

These functions apply a rank filter among 3x3 neighbors to a given image.

Usage

```
minImg(img)
maxImg(img)
```

Arguments

`img` target image

Details

In `'minImg'`, filter replaces a pixel in question with the minimum value among 3x3 neighbors. In `'maxImg'`, filter replaces a pixel in question with the maximum value among 3x3 neighbors.

Value

object of `imagematrix` class whose the size is as same as target

Examples

```
data(logo)
op <- par(mfrow=c(2,2))
plot(logo, main="Original Image") # original
plot(minImg(logo), main="Minimum-Filtered Image") # minimum filtered
plot(maxImg(logo), main="Maximum-Filtered Image") # maximum filtered
par(op)
```

read.jpeg	<i>Read JPEG file</i>
-----------	-----------------------

Description

This function reads a jpeg image file and return an imagematrix object.

Usage

```
read.jpeg(filename)
```

Arguments

filename filename of JPEG image

Value

return an imagematrix object

See Also

[imagematrix](#)

Examples

```
x <- read.jpeg(system.file("data", "cat.jpg", package="rimage"))
plot(x)
```

rgb2grey	<i>Convert color imagematrix to grey imagematrix</i>
----------	--

Description

This function convert color imagematrix to grey imagematrix.

Usage

```
rgb2grey(img, coefs=c(0.30, 0.59, 0.11))
```

Arguments

img target image
coefs coefficients for red plane, green plane, and blue plane.

Value

grey imagematrix

Examples

```
x <- read.jpeg(system.file("data", "cat.jpg", package="rimage"))
plot(rgb2grey(x))
```

`sobel`*Sobel filter*

Description

This function calculates an image which sobel filter is applied. It utilizes a C routine for improving speed.

Usage

```
sobel(img)
```

Arguments

`img` a matrix representing target image

Value

a matrix representing the image after sobel filter is applied

See Also

[sobel.h](#), [sobel.v](#), [imagematrix](#)

Examples

```
data(logo)
plot(normalize(sobel(logo)))
```

`sobel.h`*sobel filter to extract horizontal edges*

Description

This function calculates an image which sobel filter is applied. It extracts horizontal edges only. It is faster than `sobel.h` extremely because utilization of a C routine.

Usage

```
sobel.h(img)
```

Arguments

`img` a matrix representing target image

Value

a matrix representing the image after horizontal sobel filter is applied

See Also

[sobel.v](#), [sobel](#)

Examples

```
data (logo)
plot (normalize (sobel.h (logo)))
```

`sobel.v`

Sobel filter to extract vertical edges

Description

This function calculates an image which sobel filter is applied. It extracts vertical edges only. It is faster than `sobel.v` extremely because utilization of a C routine.

Usage

```
sobel.v (img)
```

Arguments

`img` a matrix representing target image

Value

a matrix representing the image after vertical sobel filter is applied

See Also

[sobel.h](#), [sobel](#)

Examples

```
data (logo)
plot (normalize (sobel.v (logo)))
```

thresholding	<i>thresholding image</i>
--------------	---------------------------

Description

This function applies thresholding to an image. You can choose fixed threshold mode or discriminial analysis mode. In fixed threshold mode, you can simply specify threshold value. In discriminial analysis mode, threshold is determined automatically so that two clusters are seperated most clearly.

Usage

```
thresholding(img, mode="fixed", th=0.5)
```

Arguments

<code>img</code>	target imagematrix image
<code>mode</code>	thresholding mode. You can specify "fixed" for fixed threshold mode or "da" for discriminial analysis mode".
<code>th</code>	threshold to be used if mode is "fixed", otherwise ignored

Value

a pixmap image

Examples

```
data(logo)
op <- par(mfrow=c(2,2))
plot(logo, main="Original")
plot(thresholding(logo, mode="fixed"), main="threshold=0.5")
plot(thresholding(logo, mode="fixed", th=0.9), main="threshold=0.9")
plot(thresholding(logo, mode="da"), main="auto threshold by discriminial analysis")
par(op)
```

Index

*Topic **datasets**

cat, 1
logo, 8

*Topic **misc**

clipping, 2
equalize, 3
fftImg, 4
fftw, 4
highpass, 5
imagematrix, 6
imageType, 7
laplacian, 8
lowpass, 9
meanImg, 9
normalize, 10
plot.imagematrix, 10
print.imagematrix, 11
Rank Filters, 12
read.jpeg, 12
rgb2grey, 13
sobel, 14
sobel.h, 14
sobel.v, 15
thresholding, 16

cat, 1
clipping, 2, 7

equalize, 3

fftImg, 4
fftw, 4, 4, 5

highpass, 5, 9

imagematrix, 4, 6, 11, 13, 14
imageType, 7

laplacian, 8
logo, 8
lowpass, 5, 9

maxImg (*Rank Filters*), 12
meanImg, 9
minImg (*Rank Filters*), 12

normalize, 7, 10

plot.imagematrix, 7, 10
print.imagematrix, 7, 11

Rank Filters, 12
read.jpeg, 2, 12
rgb2grey, 13

sobel, 14, 15
sobel.h, 14, 14, 15
sobel.v, 14, 15, 15

thresholding, 16