

Package ‘biOps’

January 24, 2012

Type Package

Title Image processing and analysis

Version 0.2.1.1

Date 2007-10-20

Author Matias Bordese, Walter Alini

Maintainer Matias Bordese <mbordese@gmail.com>

Encoding UTF-8

Description This package includes several methods for image processing and analysis. It provides geometric, arithmetic, logic, morphologic (supported on one channel images only), look-up tables, edge detection (including Roberts, Sobel, Kirsch, Marr-Hildreth and Canny, among others) and convolution masks operations (predefined commons masks already defined and user defined applications). Isodata and k-means classification methods are also provided (standard, kd-tree and brute force methods implemented). Fast Fourier Transform methods and filters also available if fftw3 installed. Supports jpeg and tiff images so far (more image support in future versions). libtiff and libjpeg libraries installed required.

SystemRequirements libjpeg; optional libtiff and fftw3

License GPL

Repository CRAN

Date/Publication 2012-01-24 16:35:32

R topics documented:

biOps-package	4
imagedata	7
imageType	9
imgAdd	9

imgAND	10
imgAverage	11
imgAverageShrink	11
imgBilinearRotate	12
imgBilinearScale	13
imgBinaryClosing	14
imgBinaryDilation	15
imgBinaryErosion	15
imgBinaryOpening	16
imgBlockMedianFilter	17
imgBlueBand	18
imgBlur	18
imgBoost	19
imgCanny	20
imgConvolve	21
imgCrop	21
imgCubicRotate	22
imgCubicScale	23
imgDecreaseContrast	24
imgDecreaseIntensity	24
imgDiffer	25
imgDifferenceEdgeDetection	26
imgDivide	27
imgEKMeans	27
imgFFT	28
imgFFTBandPass	29
imgFFTBandStop	30
imgFFTConvolve	30
imgFFTHighPass	31
imgFFTInv	32
imgFFTiShift	33
imgFFTLowPass	33
imgFFTPhase	34
imgFFTShift	35
imgFFTSpectrum	35
imgFreiChen	36
imgGamma	37
imgGaussianNoise	37
imgGetRGBFromBands	38
imgGrayScaleClosing	39
imgGrayScaleDilation	40
imgGrayScaleErosion	40
imgGrayScaleOpening	41
imgGreenBand	42
imgHighPassFilter	43
imgHistogram	44
imgHomogeneityEdgeDetection	45
imgHorizontalMirroring	46

imgIncreaseContrast	46
imgIncreaseIntensity	47
imgIsoData	48
imgKDKMeans	49
imgKirsch	49
imgKMeans	50
imgMarrHildreth	51
imgMaximum	51
imgMaximumFilter	52
imgMedianShrink	53
imgMinimumFilter	54
imgMultiply	54
imgNDilationErosion	55
imgNearestNeighborRotate	56
imgNearestNeighborScale	57
imgNegative	58
imgNErosionDilation	58
imgNormalize	59
imgOR	60
imgPadding	61
imgPrewitt	61
imgPrewittCompassGradient	62
imgRedBand	63
imgRGB2Grey	63
imgRoberts	64
imgRobinson3Level	65
imgRobinson5Level	65
imgRotate	66
imgRotate90Clockwise	67
imgRotate90CounterClockwise	68
imgSaltPepperNoise	68
imgScale	69
imgSharpen	70
imgShenCastan	70
imgSobel	71
imgSplineRotate	72
imgSplineScale	73
imgStdBinaryClosing	74
imgStdBinaryDilation	74
imgStdBinaryErosion	75
imgStdBinaryOpening	76
imgStdBlur	77
imgStdNDilationErosion	77
imgStdNErosionDilation	78
imgThreshold	79
imgTranslate	80
imgUnsharpen	80
imgVerticalMirroring	81

imgXOR	82
logo	83
plot.imagedata	83
print.imagedata	84
readJpeg	84
readTiff	85
r_dec_contrast	86
r_dec_intensity	87
r_gamma	88
r_imgAdd	89
r_imgAverage	89
r_imgDiffer	90
r_imgMaximum	91
r_inc_contrast	92
r_inc_intensity	93
r_look_up_table	94
r_negative	94
r_negative_lut	95
r_threshold	96
writeJpeg	96
writeTiff	97

Index 98

biOps-package	<i>Basic image operations and image processing</i>
---------------	--

Description

This package includes arithmetic, logic, look up table and geometric operations. Some image processing functions, for edge detection (several algorithms including roberts, sobel, kirsch, marr-hildreth, canny) and operations by convolution masks (with predefined as well as user defined masks) are provided. Supported file formats are jpeg and tiff (it requires libtiff and libjpeg libraries installed).

Details

Package:	biOps
Type:	Package
Version:	0.2
Date:	2007-08-02
Encoding:	UTF-8
SystemRequirements:	libtiff, libjpeg
License:	GPL
Built:	R 2.5.1; i486-pc-linux-gnu; 2007-10-10 23:35:42; unix

Index:

biOps-package	Basic image operations
imageType	Get information on color type of imagedata
imagedata	Generate an imagedata
imgAND	And two images
imgAdd	Add two images
imgAverage	Average images
imgAverageShrink	Shrink an image
imgBilinearRotate	Rotate an image
imgBilinearScale	Scale an image
imgBinaryClosing	Applies a "closing" to an image
imgBinaryDilation	Dilation of a binary image
imgBinaryErosion	Erosion of a binary image
imgBinaryOpening	Applies an "opening" to an image
imgBlockMedianFilter	Filters an image
imgBlueBand	Return the image blue band
imgBlur	Blurs an image
imgBoost	High Boosts an image
imgCanny	Canny Edge Detection Method
imgConvolve	Performs an image convolution
imgCrop	Crops an image
imgCubicRotate	Rotate an image
imgCubicScale	Scale an image
imgDecreaseContrast	Decrease contrast
imgDecreaseIntensity	Decrease intensity
imgDiffer	Substract two images
imgDifferenceEdgeDetection	Enhaces image edges
imgDivide	Divide two images
imgEKMeans	Image clustering
imgFFT	Fast Fourier Transformation of an image
imgFFTBandPass	Apply a band pass filter on a fft matrix
imgFFTBandStop	Apply a band stop filter on a fft matrix
imgFFTConvolve	Apply a convolution filter on an imagedata through fft transformation
imgFFTHighPass	Apply a high pass filter on a fft matrix
imgFFTInv	Fast Fourier Inverse Transformation to an image
imgFFTLowPass	Apply a low pass filter on a fft matrix
imgFFTPhase	Image representation of the fft matrix phase
imgFFTShift	Shift a matrix and leave top-left value in the center
imgFFTSpectrum	Image representation of the fft matrix spectrum
imgFFTiShift	Inverse of the imgFFTShift
imgFreiChen	Frei-Chen Edge Detection Method
imgGamma	Gamma correct an image
imgGaussianNoise	Add gaussian noise
imgGetRGBFromBands	Return an RGB image
imgGrayScaleClosing	Applies a "closing" to an image

imgGrayScaleDilation	Dilation of a gray scale image
imgGrayScaleErosion	Erosion of a gray scale image
imgGrayScaleOpening	Applies an "opening" to an image
imgGreenBand	Return the image green band
imgHighPassFilter	Sharpens an image
imgHistogram	Return the image histogram
imgHomogeneityEdgeDetection	Enhances image edges
imgHorizontalMirroring	Horizontal mirror an image
imgIncreaseContrast	Increase contrast
imgIncreaseIntensity	Increase intensity
imgIsoData	Image clustering
imgKDKMeans	Image clustering
imgKMeans	Image clustering
imgKirsch	Kirsch Edge Detection Method
imgMarrHildreth	Marr-Hildreth Edge Detection Method
imgMaximum	Calculates image maximum
imgMaximumFilter	Filters an image
imgMedianShrink	Shrink an image
imgMinimumFilter	Filters an image
imgMultiply	Multiply two images
imgNDilationErosion	Dilation/Erosion multiple apply
imgNErosionDilation	Erosion/Dilation multiple apply
imgNearestNeighborRotate	Rotate an image
imgNearestNeighborScale	Scale an image
imgNegative	Negate an image
imgNormalize	Normalization for vector and matrix
imgOR	Or two images
imgPadding	Pad an image to the given dimensions
imgPrewitt	Prewitt Edge Detection Method
imgPrewittCompassGradient	Prewitt Compass Gradient Edge Detection Method
imgRGB2Grey	Convert color imagedata to grey imagedata
imgRedBand	Return the image red band
imgRoberts	Roberts Edge Detection Method
imgRobinson3Level	Robinson 3-level Edge Detection Method
imgRobinson5Level	Robinson 5-level Edge Detection Method
imgRotate	Rotate an image
imgRotate90Clockwise	Rotate an image
imgRotate90CounterClockwise	Rotate an image
imgSaltPepperNoise	Add salt and pepper noise
imgScale	Scale an image
imgSharpen	Sharpens an image with selected mask
imgShenCastan	Shen-Castan Edge Detection Method

imgSobel	Sobel Edge Detection Method
imgSplineRotate	Rotate an image
imgSplineScale	Scale an image
imgStdBinaryClosing	Fixed mask binary closing
imgStdBinaryDilation	Fixed mask binary dilation
imgStdBinaryErosion	Fixed mask binary erosion
imgStdBinaryOpening	Fixed mask binary opening
imgStdBlur	Blurs an image
imgStdNDilationErosion	Fixed mask NDilationErosion
imgStdNErosionDilation	Fixed mask NErosionDilation
imgThreshold	Threshold an image
imgTranslate	Translate an image block
imgUnsharpen	Unsharpen an image with selected mask
imgVerticalMirroring	Vertical mirror an image
imgXOR	Xor two images
logo	R logo imagedata
plot.imagedata	Plotting an imagedata object
print.imagedata	Print information on a given imagedata object
r_dec_contrast	Decrease contrast
r_dec_intensity	Decrease intensity
r_gamma	Gamma correct an image
r_imgAdd	Add two images
r_imgAverage	Average images
r_imgDiffer	Subtract two images
r_imgMaximum	Images maximum
r_inc_contrast	Increase contrast
r_inc_intensity	Increase intensity
r_look_up_table	Transforms an image by a given look-up table
r_negative	Negate an image
r_negative_lut	Negate an image
r_threshold	Threshold an image
readJpeg	Read jpeg file
readTiff	Read tiff file
writeJpeg	Write jpeg file
writeTiff	Write tiff file

Author(s)

Matias Bordese, Walter Alini

Maintainer: Matias Bordese <mbordese@gmail.com>

Description

This function makes an imagedata object from a matrix. This data structure is primary data structure to represent image in biOps package.

Usage

```
imagedata(mat, type=NULL, ncol=dim(mat)[1], nrow=dim(mat)[2])
```

Arguments

mat	array, matrix or vector
type	"rgb" or "grey"
ncol	width of image
nrow	height of image

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.

Value

return an imagedata object

See Also

[plot.imagedata](#) [print.imagedata](#)

Examples

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

imageType	<i>Get information on color type of imagedata</i>
-----------	---

Description

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

Usage

```
imageType(x)
```

Arguments

x	The image
---	-----------

Value

"rgb" or "grey"

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
cat("Image Type", imageType(x))  
  
## End(Not run)
```

imgAdd	<i>Add two images</i>
--------	-----------------------

Description

This function adds two images and returns a new image.

Usage

```
imgAdd(imgdata1, imgdata2)
```

Arguments

imgdata1	The first image
imgdata2	The second image

Value

return an imagedata object

Note

To add a constant *c* to an image you can just do: `>> imgdata + c`.

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgAdd(x, x)

## End(Not run)
```

imgAND

And two images

Description

This function does a logic AND between two images and returns a new image.

Usage

```
imgAND(imgdata1, imgdata2)
```

Arguments

imgdata1	The first image
imgdata2	The second image

Value

return an imagedata object

See Also

[imgOR](#) [imgXOR](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgAND(x, x)

## End(Not run)
```

imgAverage	<i>Average images</i>
------------	-----------------------

Description

This function calculates the average of the given images and returns a new image.

Usage

```
imgAverage(imgdata_list)
```

Arguments

imgdata_list An image list

Value

return an imagedata object

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgAverage(list(x, x))  
  
## End(Not run)
```

imgAverageShrink	<i>Shrink an image</i>
------------------	------------------------

Description

This function shrinks an image using the average and returns a new image.

Usage

```
imgAverageShrink(imgdata, x_scale, y_scale)
```

Arguments

imgdata The image
x_scale The horizontal scale factor
y_scale The vertical scale factor

Value

return an imagedata object

Note

The scale factors are expected to be less than 1.

See Also

[imgMedianShrink](#) [imgNearestNeighborScale](#) [imgBilinearScale](#) [imgCubicScale](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgAverageShrink(x, 0.5, 0.5)  
  
## End(Not run)
```

<code>imgBilinearRotate</code>	<i>Rotate an image</i>
--------------------------------	------------------------

Description

This function rotates an image using bilinear interpolation and returns a new image.

Usage

```
imgBilinearRotate(imgdata, angle)
```

Arguments

<code>imgdata</code>	The image
<code>angle</code>	The clockwise deg angle to rotate

Value

return an imagedata object

See Also

[imgRotate](#) [imgNearestNeighborRotate](#) [imgCubicRotate](#) [imgSplineRotate](#) [imgRotate90Clockwise](#)
[imgRotate90CounterClockwise](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgBilinearRotate(x, 45)

## End(Not run)
```

imgBilinearScale	<i>Scale an image</i>
------------------	-----------------------

Description

This function scales an image using bilinear interpolation and returns a new image.

Usage

```
imgBilinearScale(imgdata, x_scale, y_scale)
```

Arguments

imgdata	The image
x_scale	The horizontal scale factor
y_scale	The vertical scale factor

Value

return an imagedata object

Note

The scale factors are expected to be greater than 1. To reduce an image use the minification functions instead.

See Also

[imgScale](#) [imgNearestNeighborScale](#) [imgCubicScale](#) [imgSplineScale](#) [imgMedianShrink](#) [imgAverageShrink](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgBilinearScale(x, 1.5, 1.5)

## End(Not run)
```

imgBinaryClosing	<i>Applies a "closing" to an image</i>
------------------	--

Description

This function applies an Dilation immediatly followed by a Erosion to the given image

Usage

```
imgBinaryClosing(imgdata, mask)
```

Arguments

imgdata	The image
mask	Mask to apply operation

Value

return an imagedata object

Note

This function accepts binary images only and will treat gray scale ones as binary images.

See Also

[imgBinaryErosion](#) [imgBinaryDilation](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
mat <- matrix (0, 3, 3)  
mask <- imagedata (mat, "grey", 3, 3)  
y <- imgBinaryClosing(x, mask)  
  
## End(Not run)
```

imgBinaryDilation	<i>Dilation of a binary image</i>
-------------------	-----------------------------------

Description

This function makes a dilation of a binary image with a given mask. This is, it applies the mask in every image pixel: when reached a black point, it turns every image's mask black point into black.

Usage

```
imgBinaryDilation(imgdata, mask)
```

Arguments

imgdata	The image
mask	Mask to apply operation

Value

return an imagedata object

Note

This function accepts binary images only and will treat gray scale ones as binary images.

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
mat <- matrix(0, 3, 3)  
mask <- imagedata(mat, "grey", 3, 3)  
y <- imgBinaryDilation(x, mask)  
  
## End(Not run)
```

imgBinaryErosion	<i>Erosion of a binary image</i>
------------------	----------------------------------

Description

This function makes an erosion of a binary image with a given mask. This is, it applies the mask in every image pixel: when the mask matches completely, it turns its images' center into a black point

Usage

```
imgBinaryErosion(imgdata, mask)
```

Arguments

imgdata	The image
mask	Mask to apply operation

Value

return an imagedata object

Note

This function accepts binary images only and will treat gray scale ones as binary images.

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
mat <- matrix (0, 3, 3)
mask <- imagedata (mat, "grey", 3, 3)
y <- imgBinaryErosion(x, mask)

## End(Not run)
```

imgBinaryOpening	<i>Applies an "opening" to an image</i>
------------------	---

Description

This function applies an erosion immediately followed by a dilation to the given image

Usage

```
imgBinaryOpening(imgdata, mask)
```

Arguments

imgdata	The image
mask	Mask to apply operation

Value

return an imagedata object

Note

This function accepts binary images only and will treat gray scale ones as binary images.

See Also

[imgBinaryErosion](#) [imgBinaryDilation](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
mat <- matrix(0, 3, 3)
mask <- imagedata(mat, "grey", 3, 3)
y <- imgBinaryOpening(x, mask)

## End(Not run)
```

imgBlockMedianFilter *Filters an image*

Description

This function filters an image by the Median filter, with a block window with a given dimension

Usage

```
imgBlockMedianFilter (imgdata, dim)
```

Arguments

imgdata	The image
dim	Block's dimension (default=3)

Value

return an imagedata object

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgBlockMedianFilter(x, 5)

## End(Not run)
```

imgBlueBand	<i>Return the image blue band</i>
-------------	-----------------------------------

Description

This function returns the blue band of the imagedata.

Usage

```
imgBlueBand(x)
```

Arguments

x	The image
---	-----------

Value

grey imagedata

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
plot(imgBlueBand(x))  
  
## End(Not run)
```

imgBlur	<i>Blurs an image</i>
---------	-----------------------

Description

This function blurs an image by convoluting with the following matrix:

$$\begin{matrix} 1/16 & 1/8 & 1/16 \\ 1/8 & 1/4 & 1/8 \\ 1/16 & 1/8 & 1/16 \end{matrix}$$
Usage

```
imgBlur(imgdata)
```

Arguments

imgdata	The image
---------	-----------

Value

return an imagedata object

See Also

[imgStdBlur](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgStdBlur(x)  
  
## End(Not run)
```

imgBoost

High Boosts an image

Description

This function high boosts an image by convoluting with the following matrix:

$$\begin{matrix} -1/9 & -1/9 & -1/9 \\ -1/9 & (9p-1)/9 & -1/9 \\ -1/9 & -1/9 & -1/9 \end{matrix}$$

It increases intensity by a given proportion (p) and subtracting a lowpass filter

Usage

```
imgBoost(imgdata, proportion)
```

Arguments

imgdata The image
proportion Proportion of intensity to be increased (optional: default = 1 -HighPassFilter-)

Value

return an imagedata object

Note

When proportion=1, it's the same as [imgHighPassFilter](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgBoost(x, 1.2)  
  
## End(Not run)
```

imgCanny

Canny Edge Detection Method

Description

This function does edge detection using the Canny algorithm.

Usage

```
imgCanny(imgdata, sigma, low=0, high=-1)
```

Arguments

imgdata	The image
sigma	The standard deviation used for the gaussian smoothing convolution
low	The lower threshold for hysteresis
high	The higher threshold for hysteresis

Value

return an imagedata object

Note

If not specified, the low and high parameters are estimated based in a histogram of the image.

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgCanny(x, 0.7)  
  
## End(Not run)
```

imgConvolve	<i>Performs an image convolution</i>
-------------	--------------------------------------

Description

This function performs an image convolution with given mask

Usage

```
imgConvolve(imgdata, mask, bias)
```

Arguments

imgdata	The image
mask	Kernel's convolution matrix
bias	Value to be added to each pixel after method is applied (used to correct some expected behaviour). This argument is optional (default = 32)

Value

return an imagedata object

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
m <- matrix(c(1,2,1,2,4,2,1,2,1)/16, 3, 3, byrow = TRUE)  
y <- imgConvolve(x, m, 64)  
  
## End(Not run)
```

imgCrop	<i>Crops an image</i>
---------	-----------------------

Description

This function crops image.

Usage

```
imgCrop(imgdata, x_start, y_start, c_width, c_height)
```

Arguments

imgdata	The image
x_start	Upper left x coordinate of source block
y_start	Upper left y coordinate of source block
c_width	Width of the block to crop
c_height	Height of the block to crop

Value

return an imagedata object

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgCrop(x, 100, 50, 100, 50)  
  
## End(Not run)
```

imgCubicRotate	<i>Rotate an image</i>
----------------	------------------------

Description

This function rotates an image using cubic interpolation and returns a new image.

Usage

```
imgCubicRotate(imgdata, angle)
```

Arguments

imgdata	The image
angle	The clockwise deg angle to rotate

Value

return an imagedata object

See Also

[imgRotate](#) [imgNearestNeighborRotate](#) [imgBilinearRotate](#) [imgSplineRotate](#) [imgRotate90Clockwise](#)
[imgRotate90CounterClockwise](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgCubicRotate(x, 45)

## End(Not run)
```

imgCubicScale	<i>Scale an image</i>
---------------	-----------------------

Description

This function scales an image using cubic interpolation and returns a new image.

Usage

```
imgCubicScale(imgdata, x_scale, y_scale)
```

Arguments

imgdata	The image
x_scale	The horizontal scale factor
y_scale	The vertical scale factor

Value

return an imagedata object

Note

The scale factors are expected to be greater than 1. To reduce an image use the minification functions instead.

See Also

[imgScale](#) [imgNearestNeighborScale](#) [imgBilinearScale](#) [imgSplineScale](#) [imgMedianShrink](#)
[imgAverageShrink](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgCubicScale(x, 1.5, 1.5)

## End(Not run)
```

`imgDecreaseContrast` *Decrease contrast*

Description

This function decreases an image contrast, leaving each pixel value between given values.

Usage

```
imgDecreaseContrast(imgdata, min_desired, max_desired)
```

Arguments

<code>imgdata</code>	The image
<code>min_desired</code>	The min value
<code>max_desired</code>	The max value

Value

return an imagedata object

See Also

[imgIncreaseContrast](#) [r_dec_contrast](#) [r_inc_contrast](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgDecreaseContrast(x, 60, 200)  
  
## End(Not run)
```

`imgDecreaseIntensity` *Decrease intensity*

Description

This function decreases an image intensity by a given factor.

Usage

```
imgDecreaseIntensity(imgdata, percentage)
```

Arguments

imgdata	The image
percentage	A non negative value representing the intensity percentage to be decreased. 1 stands for 100% (eg. 0.5 = 50%).

Value

return an imagedata object

See Also

[imgIncreaseIntensity](#) [r_dec_intensity](#) [r_inc_intensity](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgDecreaseIntensity(x, 0.3)  
  
## End(Not run)
```

imgDiffer	<i>Subtract two images</i>
-----------	----------------------------

Description

This function subtracts two images and returns a new image, `imgdata1 - imgdata2`.

Usage

```
imgDiffer(imgdata1, imgdata2)
```

Arguments

imgdata1	The first image
imgdata2	The second image

Value

return an imagedata object

Note

To subtract a constant `c` to an image you can just do: `>> imgdata - c`.

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgDiffer(x, x)  
  
## End(Not run)
```

imgDifferenceEdgeDetection
Enhances image edges

Description

This function enhances image's edge by the difference method. It uses a 3x3 matrix to determine the current pixel value (by getting the maximum value between the distances of matrix's opposite neighbors)

Usage

```
imgDifferenceEdgeDetection(imgdata, bias)
```

Arguments

imgdata	The image
bias	Value to be added to each pixel after method is applied (used to correct some expected behaviour). This argument is optional (default = 32)

Value

return an imagedata object

See Also

[imgHomogeneityEdgeDetection](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgDifferenceEdgeDetection(x, bias=64)  
  
## End(Not run)
```

imgDivide	<i>Divide two images</i>
-----------	--------------------------

Description

This function divides two images and returns a new image.

Usage

```
imgDivide(imgdata1, imgdata2)
```

Arguments

imgdata1	The first image
imgdata2	The second image

Value

return an imagedata object

Note

To divide an image by a constant *c* you can just do: `>> imgdata / c.`

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgDivide(x, x)  
  
## End(Not run)
```

imgEKMeans	<i>Image clustering</i>
------------	-------------------------

Description

This function performs an unsupervised classification through the k-means algorithm. It is an enhanced implementation, that avoid some comparisons based on kept information about distances and centroids of previous iterations.

Usage

```
imgEKMeans (imgdata, k, maxit=10)
```

Arguments

imgdata	The image
k	Number of clusters
maxit	Max number of iterations

Value

return an imagedata object, the result of the classification

See Also

[imgKMeans](#) [imgKDKMeans](#) [imgIsoData](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgEKMeans(x, 4)  
  
## End(Not run)
```

imgFFT

Fast Fourier Transformation of an image

Description

This function applies a Fast Fourier Transformation on an imagedata.

Usage

```
imgFFT(imgdata, shift = TRUE)
```

Arguments

imgdata	The image
shift	If TRUE (default), the transformation origin is centered

Value

return a complex matrix

See Also

[imgFFTInv](#) [imgFFTShift](#) [imgFFTiShift](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
t <- imgFFT(x)  
  
## End(Not run)
```

imgFFTBandPass	<i>Apply a band pass filter on a fft matrix</i>
----------------	---

Description

This function returns the band passed filter on a fft matrix (this matrix should be shifted).

Usage

```
imgFFTBandPass(fft_matrix, r1, r2)
```

Arguments

fft_matrix	The complex matrix of an fft transformation
r1	The inner radius of the frequency filter
r2	The outer radius of the frequency filter

Value

return an imagedata

See Also

[imgFFT](#) [imgFFTInv](#) [imgFFTLowPass](#) [imgFFTHighPass](#) [imgFFTBandStop](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
t <- imgFFT(x, FALSE)  
i <- imgFFTBandPass(t, 25, 70)  
  
## End(Not run)
```

imgFFTBandStop	<i>Apply a band stop filter on a fft matrix</i>
----------------	---

Description

This function returns the band stop filter on a fft matrix (this matrix should be shifted).

Usage

```
imgFFTBandStop(fft_matrix, r1, r2)
```

Arguments

fft_matrix	The complex matrix of an fft transformation
r1	The inner radius of the frequency filter
r2	The outer radius of the frequency filter

Value

return an imagedata

See Also

[imgFFT](#) [imgFFTInv](#) [imgFFTLowPass](#) [imgFFTHighPass](#) [imgFFTBandPass](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
t <- imgFFT(x, FALSE)  
i <- imgFFTBandStop(t, 25, 70)  
  
## End(Not run)
```

imgFFTConvolve	<i>Apply a convolution filter on an imagedata through fft transformation</i>
----------------	--

Description

This function returns the imagedata that results from the convolution, using fft transformation that let you convolve with bigger masks.

Usage

```
imgFFTConvolve(imgdata, mask)
```

Arguments

imgdata The image
mask The convolution mask

Value

return an imagedata

See Also

[imgFFT](#) [imgFFTIInv](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
data <- c(-1,-1,-1,-1,9,-1,-1,-1,-1)  
m <- matrix(data, 3, 3, byrow = TRUE)  
i <- imgFFTConvolve(x, m)  
  
## End(Not run)
```

imgFFTHighPass *Apply a high pass filter on a fft matrix*

Description

This function returns the high passed filter on a fft matrix (this matrix should be shifted).

Usage

```
imgFFTHighPass(fft_matrix, r)
```

Arguments

fft_matrix The complex matrix of an fft transformation
r The radius of the frequency filter

Value

return an imagedata

See Also

[imgFFT](#) [imgFFTIInv](#) [imgFFTLowPass](#) [imgFFTBandPass](#) [imgFFTBandStop](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
t <- imgFFT(x, FALSE)
i <- imgFFTHighPass(t, 25)

## End(Not run)
```

imgFFTInv

Fast Fourier Inverse Transformation to an image

Description

This function applies a Fast Fourier Inverse Transformation on a complex matrix and return an imagedata.

Usage

```
imgFFTInv(fft_matrix, shift = TRUE)
```

Arguments

fft_matrix	The image
shift	If TRUE, the transformation origin is moved to the top-left before the inverse

Value

return an imagedata

See Also

[imgFFT](#) [imgFFTShift](#) [imgFFTiShift](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
t <- imgFFT(x)
i <- imgFFTInv(t)

## End(Not run)
```

imgFFTiShift	<i>Inverse of the imgFFTShift</i>
--------------	-----------------------------------

Description

This function returns the inverse shifted matrix, useful in fft transformation.

Usage

```
imgFFTiShift(imgmatrix)
```

Arguments

`imgmatrix` A matrix (could be an image or a fft matrix)

Value

return a matrix

See Also

[imgFFT](#) [imgFFTIInv](#) [imgFFTShift](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
t <- imgFFTShift(x)  
i <- imgFFTiShift(t)  
  
## End(Not run)
```

imgFFTLowPass	<i>Apply a low pass filter on a fft matrix</i>
---------------	--

Description

This function returns the low passed filter on a fft matrix (this matrix should be shifted).

Usage

```
imgFFTLowPass(fft_matrix, r)
```

Arguments

`fft_matrix` The complex matrix of an fft transformation
`r` The radius of the frequency filter

Value

return an imagedata

See Also

[imgFFT](#) [imgFFTIInv](#) [imgFFTHighPass](#) [imgFFTBandPass](#) [imgFFTBandStop](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
t <- imgFFT(x, FALSE)
i <- imgFFTLowPass(t, 25)

## End(Not run)
```

imgFFTPPhase

Image representation of the fft matrix phase

Description

This function returns the respective phase of the given complex matrix (ie the result of a fft transformation).

Usage

```
imgFFTPPhase(fft_matrix)
```

Arguments

`fft_matrix` The complex matrix of an fft transformation

Value

return an imagedata

See Also

[imgFFT](#) [imgFFTIInv](#) [imgFFTSpectrum](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
t <- imgFFT(x)
i <- imgFFTPPhase(t)

## End(Not run)
```

imgFFTShift	<i>Shift a matrix and leave top-left value in the center</i>
-------------	--

Description

This function returns the shifted matrix, useful in fft transformation.

Usage

```
imgFFTShift(imgmatrix)
```

Arguments

`imgmatrix` A matrix (could be an image or a fft matrix)

Value

return a matrix

See Also

[imgFFT](#) [imgFFTInv](#) [imgFFTiShift](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
t <- imgFFT(x, FALSE)
i <- imgFFTShift(t)

## End(Not run)
```

imgFFTSpectrum	<i>Image representation of the fft matrix spectrum</i>
----------------	--

Description

This function returns the respective spectrum of the given complex matrix (ie the result of a fft transformation).

Usage

```
imgFFTSpectrum(fft_matrix)
```

Arguments

`fft_matrix` The complex matrix of an fft transformation

Value

return an imagedata

See Also

[imgFFT](#) [imgFFTIInv](#) [imgFFTPhase](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
t <- imgFFT(x)
i <- imgFFTSpectrum(t)

## End(Not run)
```

Frei-Chen Edge Detection Method

Description

This function enhances image's edges by convoluting with the Frei-Chen method matrices:

$$\begin{array}{cccccc}
 & \mathbf{H_r} & & & \mathbf{H_c} & \\
 1 & 0 & -1 & \parallel & -1 & -\sqrt{2} & -1 \\
 \sqrt{2} & 0 & -\sqrt{2} & \parallel & 0 & 0 & 0 \\
 1 & 0 & -1 & \parallel & 1 & \sqrt{2} & 1
 \end{array}$$

Usage

```
imgFreiChen(imgdata)
```

Arguments

imgdata The image

Value

return an imagedata object

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgFreiChen(x)

## End(Not run)
```

imgGamma	<i>Gamma correct an image</i>
----------	-------------------------------

Description

This function applies gamma operation to a given image. Each pixel value is taken to the inverse of gamma_value-th exponent.

Usage

```
imgGamma(imgdata, gamma_value)
```

Arguments

imgdata	The image
gamma_value	A non negative value representing operation gamma value

Value

return an imagedata object

See Also

[r_gamma](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgGamma(x, 1.3)  
  
## End(Not run)
```

imgGaussianNoise	<i>Add gaussian noise</i>
------------------	---------------------------

Description

This function adds gaussian noise to an image.

Usage

```
imgGaussianNoise(imgdata, mean, variance)
```

Arguments

imgdata	The image
mean	The gaussian mean
variance	The gaussian variance

Value

return an imagedata object

See Also

[imgSaltPepperNoise](#)

Examples

```
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgGaussianNoise(x, 0, 120)
```

imgGetRGBFromBands *Return an RGB image*

Description

This function returns the RGB image compositing the given bands.

Usage

```
imgGetRGBFromBands(R, G, B)
```

Arguments

R	A one-band image for the Red band
G	A one-band image for the Green band
B	A one-band image for the Blue band

Value

RGB imagedata

Examples

```
## Not run: x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
r <- imgRedBand(x)
g <- imgGreenBand(x)
b <- imgBlueBand(x)
rgb <- imgGetRGBFromBands(r, g, b)

## End(Not run)
```

imgGrayScaleClosing *Applies a “closing” to an image*

Description

This function applies an Dilation immediatly followed by a Erosion to the given image

Usage

```
imgGrayScaleClosing(imgdata, mask)
```

Arguments

imgdata	The image
mask	Mask to apply operation

Value

return an imagedata object

Note

This function accepts gray scale images and will fail with color ones.

See Also

[imgGrayScaleErosion](#) [imgGrayScaleDilation](#) [imgBinaryErosion](#) [imgBinaryDilation](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
mat <- matrix (0, 3, 3)  
mask <- imagedata (mat, "grey", 3, 3)  
y <- imgGrayScaleClosing(x, mask)  
  
## End(Not run)
```

imgGrayScaleDilation *Dilation of a gray scale image*

Description

This function makes a dilation of a gray scale image with a given mask. This is, it applies the mask in every image pixel and sets current point to the maximum of the sums of the corresponding pair of pixel values in the mask and image.

Usage

```
imgGrayScaleDilation(imgdata, mask)
```

Arguments

imgdata	The image
mask	Mask to apply operation

Value

return an imagedata object

Note

This function accepts gray scale images and will fail with color ones.

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
mat <- matrix(0, 3, 3)  
mask <- imagedata(mat, "grey", 3, 3)  
y <- imgGrayScaleDilation(x, mask)  
  
## End(Not run)
```

imgGrayScaleErosion *Erosion of a gray scale image*

Description

This function makes a dilation of a gray scale image with a given mask. This is, it applies the mask in every image pixel and sets current point to the minimum of the sums of the corresponding pair of pixel values in the mask and image.

Usage

```
imgGrayScaleErosion(imgdata, mask)
```

Arguments

imgdata	The image
mask	Mask to apply operation

Value

return an imagedata object

Note

This function accepts gray scale images and will fail with color ones.

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
mat <- matrix (0, 3, 3)  
mask <- imagedata (mat, "grey", 3, 3)  
y <- imgGrayScaleErosion(x, mask)  
  
## End(Not run)
```

imgGrayScaleOpening *Applies an "opening" to an image*

Description

This function applies an erosion immediately followed by a dilation to the given image

Usage

```
imgGrayScaleOpening(imgdata, mask)
```

Arguments

imgdata	The image
mask	Mask to apply operation

Value

return an imagedata object

Note

This function accepts gray scale images and will fail with color ones.

See Also

[imgGrayScaleErosion](#) [imgGrayScaleDilation](#) [imgBinaryErosion](#) [imgBinaryDilation](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
mat <- matrix (125, 3, 3)
mask <- imagedata (mat, "grey", 3, 3)
y <- imgGrayScaleOpening(x, mask)

## End(Not run)
```

imgGreenBand	<i>Return the image green band</i>
--------------	------------------------------------

Description

This function returns the green band of the imagedata.

Usage

```
imgGreenBand(x)
```

Arguments

x The image

Value

grey imagedata

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
plot(imgGreenBand(x))

## End(Not run)
```

`imgHighPassFilter` *Sharpens an image*

Description

This function sharpens an image by convoluting with the following matrix:

```
-1/9 -1/9 -1/9
-1/9 8/9 -1/9
-1/9 -1/9 -1/9
```

Usage

```
imgHighPassFilter (imgdata)
```

Arguments

```
imgdata      The image
```

Value

return an imagedata object

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgHighPassFilter(x)

## End(Not run)
```

```
imgHistogram      Return the image histogram
```

Description

This function returns the image pixel values histogram.

Usage

```
imgHistogram(x, main='Image Histogram', col='Midnight Blue', ...)
```

Arguments

```
x              The image
main           The histogram title
col            The histogram bars color
...           Same options of hist function
```

Value

histogram object

See Also[hist](#)**Examples**

```
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
h <- imgHistogram(x)
```

imgHomogeneityEdgeDetection
Enhances image edges

Description

This function enhances image's edge by the homogeneity method. It uses a 3x3 matrix to determine the current pixel value (by getting the maximum value between the distances of the pixel and its neighbors)

Usage

```
imgHomogeneityEdgeDetection(imgdata, bias)
```

Arguments

imgdata	The image
bias	Value to be added to each pixel after method is applied (used to correct some expected behaviour). This argument is optional (default = 32)

Value

return an imagedata object

See Also[imgHomogeneityEdgeDetection](#)**Examples**

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgHomogeneityEdgeDetection(x, bias=64)

## End(Not run)
```

`imgHorizontalMirroring`*Horizontal mirror an image*

Description

This function flips an image about the y axis.

Usage

```
imgHorizontalMirroring(imgdata)
```

Arguments

`imgdata` The image

Value

return an imagedata object

See Also

[imgVerticalMirroring](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgHorizontalMirroring(x)  
  
## End(Not run)
```

`imgIncreaseContrast` *Increase contrast*

Description

This function increases an image contrast, augmenting pixel values differences between given limits (in a linear fashion).

Usage

```
imgIncreaseContrast(imgdata, min_limit, max_limit)
```

Arguments

<code>imgdata</code>	The image
<code>min_limit</code>	The minimum limit to apply lineal modification
<code>max_limit</code>	The maximum limit to apply lineal modification

Value

return an imagedata object

See Also

[imgDecreaseContrast](#) [r_inc_contrast](#) [r_dec_contrast](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgIncreaseContrast(x, 60, 200)  
  
## End(Not run)
```

`imgIncreaseIntensity` *Increase intensity*

Description

This function increases an image intensity by a given factor.

Usage

```
imgIncreaseIntensity(imgdata, percentage)
```

Arguments

<code>imgdata</code>	The image
<code>percentage</code>	A non negative value representing the intensity percentage to be increased. 1 stands for 100% (eg. 0.5 = 50%)

Value

return an imagedata object

See Also

[imgDecreaseIntensity](#) [r_inc_intensity](#) [r_dec_intensity](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgIncreaseIntensity(x, 0.3)

## End(Not run)
```

imgIsoData

Image clustering

Description

This function performs an unsupervised classification through the k-means algorithm. It is an enhanced implementation, that avoid some comparisons based on kept information about distances and centroids of previous iterations.

Usage

```
imgIsoData (imgdata, k, min_dist=1, min_elems=1, split_sd=0.1, iter_start=5, max_merge=2, max_iter=10)
```

Arguments

imgdata	The image
k	Number of clusters
min_dist	Minimum distance between cluster centroids
min_elems	Minimum elements per cluster
split_sd	Standard deviation threshold for splitting operation
iter_start	Maximum number of forgy iterations
max_merge	Maximum of merge operations per iteration
max_iter	Maximum number of iterations

Value

return an imagedata object, the result of the classification

See Also

[imgKMeans](#) [imgEKMeans](#) [imgKDKMeans](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgIsoData(x, 4)

## End(Not run)
```

imgKDKMeans	<i>Image clustering</i>
-------------	-------------------------

Description

This function performs an unsupervised classification through the k-means algorithm. This implementation uses kd-trees for nearest neighbor queries. It is useful for big values of k.

Usage

```
imgKDKMeans (imgdata, k, maxit=10)
```

Arguments

imgdata	The image
k	Number of clusters
maxit	Max number of iterations

Value

return an imagedata object, the result of the classification

See Also

[imgKMeans](#) [imgEKMeans](#) [imgIsoData](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgKDKMeans(x, 4)  
  
## End(Not run)
```

imgKirsch	<i>Kirsch Edge Detection Method</i>
-----------	-------------------------------------

Description

This function enhances image's edges by convoluting with the Kirsch method. Base matrix is:

$$\begin{matrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{matrix}$$

Usage

```
imgKirsch(imgdata)
```

Arguments

```
imgdata      The image
```

Value

return an imagedata object

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgKirsch(x)  
  
## End(Not run)
```

imgKMeans

Image clustering

Description

This function performs an unsupervised classification through the k-means algorithm. It is a straightforward implementation.

Usage

```
imgKMeans (imgdata, k, maxit=10)
```

Arguments

```
imgdata      The image  
k            Number of clusters  
maxit       Max number of iterations
```

Value

return an imagedata object, the result of the classification

See Also

[imgEKMeans](#) [imgKDKMeans](#) [imgIsoData](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgKMeans(x, 4)

## End(Not run)
```

imgMarrHildreth	<i>Marr-Hildreth Edge Detection Method</i>
-----------------	--

Description

This function does edge detection using the Marr-Hildreth algorithm.

Usage

```
imgMarrHildreth(imgdata, sigma)
```

Arguments

imgdata	The image
sigma	The standard deviation of Gaussian for convolution

Value

return an imagedata object

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgMarrHildreth(x, 2)

## End(Not run)
```

imgMaximum	<i>Calculates image maximum</i>
------------	---------------------------------

Description

This function calculates the maximum of the given images and returns a new image.

Usage

```
imgMaximum(imgdata_list)
```

Arguments

imgdata_list An image list

Value

return an imagedata object

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgMaximum(list(x, x))  
  
## End(Not run)
```

imgMaximumFilter *Filters an image*

Description

This function filters an image by the Maximum filter, with a block window with a given dimension

Usage

```
imgMaximumFilter (imgdata, dim)
```

Arguments

imgdata The image
dim Block's dimension (default=3)

Value

return an imagedata object

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgMaximumFilter(x, 5)  
  
## End(Not run)
```

imgMedianShrink	<i>Shrink an image</i>
-----------------	------------------------

Description

This function shrinks an image using the median and returns a new image.

Usage

```
imgMedianShrink(imgdata, x_scale, y_scale)
```

Arguments

imgdata	The image
x_scale	The horizontal scale factor
y_scale	The vertical scale factor

Value

return an imagedata object

Note

The scale factors are expected to be less than 1.

See Also

[imgAverageShrink](#) [imgNearestNeighborScale](#) [imgBilinearScale](#) [imgCubicScale](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgMedianShrink(x, 0.5, 0.5)  
  
## End(Not run)
```

imgMinimumFilter *Filters an image*

Description

This function filters an image by the Minimum filter, with a block window with a given dimension

Usage

```
imgMinimumFilter (imgdata, dim)
```

Arguments

imgdata	The image
dim	Block's dimension (default=3)

Value

return an imagedata object

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgMinimumFilter(x, 5)  
  
## End(Not run)
```

imgMultiply *Multiply two images*

Description

This function multiplies two images and returns a new image.

Usage

```
imgMultiply(imgdata1, imgdata2)
```

Arguments

imgdata1	The first image
imgdata2	The second image

Value

return an imagedata object

Note

To multiply an image by a constant *c* you can just do: `>> imgdata * c.`

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgMultiply(x, x)  
  
## End(Not run)
```

imgNDilationErosion *Dilation/Erosion multiple apply*

Description

This function applies *n* dilations followed by *n* erosions to the given image. Smoothes of irregularities of *N* pixels in size

Usage

```
imgNDilationErosion(imgdata, mask, n)
```

Arguments

imgdata	The image
mask	Mask to apply operation
n	Times to apply each operation

Value

return an imagedata object

Note

This function accepts binary images only and will treat gray scale ones as binary images.

See Also

[imgBinaryErosion](#) [imgBinaryDilation](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
mat <- matrix (0, dim, dim)
mask <- imagedata (mat, "grey", dim, dim)
y <- imgNDilationErosion(x, mask, 5)

## End(Not run)
```

imgNearestNeighborRotate

Rotate an image

Description

This function rotates an image using nearest neighbor interpolation and returns a new image.

Usage

```
imgNearestNeighborRotate(imgdata, angle)
```

Arguments

imgdata	The image
angle	The clockwise deg angle to rotate

Value

return an imagedata object

See Also

[imgRotate](#) [imgBilinearRotate](#) [imgCubicRotate](#) [imgSplineRotate](#) [imgRotate90Clockwise](#) [imgRotate90CounterClockwise](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgNearestNeighborRotate(x, 45)

## End(Not run)
```

imgNearestNeighborScale
Scale an image

Description

This function scales an image using nearest neighbor interpolation and returns a new image.

Usage

```
imgNearestNeighborScale(imgdata, x_scale, y_scale)
```

Arguments

imgdata	The image
x_scale	The horizontal scale factor
y_scale	The vertical scale factor

Value

return an imagedata object

Note

The scale factors are expected to be greater than 1. To reduce an image use the minification functions instead.

See Also

[imgScale](#) [imgBilinearScale](#) [imgCubicScale](#) [imgSplineScale](#) [imgMedianShrink](#) [imgAverageShrink](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgNearestNeighborScale(x, 1.5, 1.5)  
  
## End(Not run)
```

imgNegative *Negate an image*

Description

This function negates an image.

Usage

```
imgNegative(imgdata)
```

Arguments

imgdata The image

Value

return an imagedata object

See Also

[r_negative](#) [r_negative_lut](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgNegative(x)  
  
## End(Not run)
```

imgNErosionDilation *Erosion/Dilation multiple apply*

Description

This function applies n erosions followed by n dilations to the given image. Approaches an N depth opening

Usage

```
imgNErosionDilation(imgdata, mask, n)
```

Arguments

imgdata	The image
mask	Mask to apply operation
n	Times to apply each operation

Value

return an imagedata object

Note

This function accepts binary images only and will treat gray scale ones as binary images.

See Also

[imgBinaryErosion](#) [imgBinaryDilation](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
mat <- matrix(0, dim, dim)  
mask <- imagedata(mat, "grey", dim, dim)  
y <- imgNErosionDilation(x, mask, 5)  
  
## End(Not run)
```

imgNormalize

Normalization for vector and matrix

Description

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

Usage

```
imgNormalize(x)
```

Arguments

x	The image
---	-----------

Value

Data of the same type as 'x', in which minimum value is 0 and maximum value is 255.

Examples

```
## Not run:  
data(logo)  
plot(imgNormalize(logo))  
  
## End(Not run)
```

imgOR

Or two images

Description

This function does a logic OR between two images and returns a new image.

Usage

```
imgOR(imgdata1, imgdata2)
```

Arguments

imgdata1	The first image
imgdata2	The second image

Value

return an imagedata object

See Also

[imgAND](#) [imgXOR](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgOR(x, x)  
  
## End(Not run)
```

imgPadding *Pad an image to the given dimensions*

Description

This function returns an imagedata padded to the given dimensions, leaving the input imagedata in the center of the result.

Usage

```
imgPadding(imgdata, n, m = n)
```

Arguments

imgdata	The image
n	The new width
m	The new height

Value

return an imagedata

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
t <- imgPadding(x, 350)

## End(Not run)
```

imgPrewitt *Prewitt Edge Detection Method*

Description

This function enhances image's edges by convoluting with the Prewitt method matrices:

	H_r				H_c		
1	0	-1		-1	-1	-1	
1	0	-1		0	0	0	
1	0	-1		1	1	1	

Usage

```
imgPrewitt(imgdata)
```

Arguments

```
imgdata      The image
```

Value

return an imagedata object

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgPrewitt(x)  
  
## End(Not run)
```

```
imgPrewittCompassGradient
```

Prewitt Compass Gradient Edge Detection Method

Description

This function enhances image's edges by convoluting with the Prewitt method. Base matrix is:

$$\begin{matrix} 1 & 1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & -1 \end{matrix}$$
Usage

```
imgPrewittCompassGradient(imgdata)
```

Arguments

```
imgdata      The image
```

Value

return an imagedata object

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgPrewittCompassGradient(x)
```

```
## End(Not run)
```

imgRedBand	<i>Return the image red band</i>
------------	----------------------------------

Description

This function returns the red band of the imagedata.

Usage

```
imgRedBand(x)
```

Arguments

x	The image
---	-----------

Value

grey imagedata

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
plot(imgRedBand(x))  
  
## End(Not run)
```

imgRGB2Grey	<i>Convert color imagedata to grey imagedata</i>
-------------	--

Description

This function convert color imagedata to grey imagedata.

Usage

```
imgRGB2Grey(x, coefs=c(0.30, 0.59, 0.11))
```

Arguments

x	The image
coefs	The coefficients for red, green and blue bands

Value

grey imagedata

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
plot(imgRGB2Grey(x))

## End(Not run)
```

imgRoberts

Roberts Edge Detection Method

Description

This function enhances image's edges by convoluting with the Roberts method matrices:

$$\begin{array}{cccccc}
 & & \mathbf{H_r} & & & \mathbf{H_c} & \\
 0 & 0 & -1 & \parallel & -1 & 0 & 0 \\
 0 & 1 & 0 & \parallel & 0 & 1 & 0 \\
 0 & 0 & 0 & \parallel & 0 & 0 & 0
 \end{array}$$

Usage

```
imgRoberts(imgdata)
```

Arguments

imgdata The image

Value

return an imagedata object

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgRoberts(x)

## End(Not run)
```

imgRobinson3Level *Robinson 3-level Edge Detection Method*

Description

This function enhances image's edges by convoluting with the Robinson 3-level method. Base matrix is:

$$\begin{matrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{matrix}$$
Usage

```
imgRobinson3Level(imgdata)
```

Arguments

imgdata The image

Value

return an imagedata object

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgRobinson3Level(x)

## End(Not run)
```

imgRobinson5Level *Robinson 5-level Edge Detection Method*

Description

This function enhances image's edges by convoluting with the Robinson 5-level method. Base matrix is:

$$\begin{matrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{matrix}$$

Usage

```
imgRobinson5Level(imgdata)
```

Arguments

```
imgdata      The image
```

Value

return an imagedata object

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgRobinson5Level(x)  
  
## End(Not run)
```

imgRotate

Rotate an image

Description

This function rotates an image using the given interpolation and returns a new image.

Usage

```
imgRotate(imgdata, angle, interpolation)
```

Arguments

```
imgdata      The image  
angle        The clockwise deg angle to rotate  
interpolation The interpolation method: nearestneighbor | bilinear | cubic | spline
```

Value

return an imagedata object

See Also

[imgNearestNeighborRotate](#) [imgBilinearRotate](#) [imgCubicRotate](#) [imgSplineRotate](#) [imgRotate90Clockwise](#)
[imgRotate90CounterClockwise](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgRotate(x, 45, 'spline')  
  
## End(Not run)
```

imgRotate90Clockwise *Rotate an image*

Description

This function rotates the image 90 degrees clockwise.

Usage

```
imgRotate90Clockwise(imgdata)
```

Arguments

`imgdata` The image

Value

return an imagedata object

See Also

[imgRotate90CounterClockwise](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgRotate90Clockwise(x)  
  
## End(Not run)
```

imgRotate90CounterClockwise
Rotate an image

Description

This function rotates the image 90 degrees counter-clockwise.

Usage

```
imgRotate90CounterClockwise(imgdata)
```

Arguments

imgdata The image

Value

return an imagedata object

See Also

[imgRotate90Clockwise](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgRotate90CounterClockwise(x)  
  
## End(Not run)
```

imgSaltPepperNoise *Add salt and pepper noise*

Description

This function adds salt and pepper noise to an image.

Usage

```
imgSaltPepperNoise(imgdata, percent)
```

Arguments

imgdata The image
percent The percent of noise to add

Value

return an imagedata object

See Also

[imgGaussianNoise](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgSaltPepperNoise(x, 30)  
  
## End(Not run)
```

imgScale

Scale an image

Description

This function scales an image using the given interpolation and returns a new image.

Usage

```
imgScale(imgdata, x_scale, y_scale, interpolation)
```

Arguments

imgdata	The image
x_scale	The horizontal scale factor
y_scale	The vertical scale factor
interpolation	The interpolation method: nearestneighbor bilinear cubic spline

Value

return an imagedata object

Note

The scale factors are expected to be greater than 1. To reduce an image use the minification functions instead.

See Also

[imgNearestNeighborScale](#) [imgBilinearScale](#) [imgCubicScale](#) [imgSplineScale](#) [imgMedianShrink](#)
[imgAverageShrink](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgScale(x, 1.5, 1.5, 'bilinear')

## End(Not run)
```

imgSharpen

*Sharpens an image with selected mask***Description**

This function sharpens an image by convoluting with one of the following matrices:

	1				2				3	
0	-1	0		-1	-1	-1		1	-2	1
-1	5	-1		-1	9	-1		-2	5	-2
0	-1	0		-1	-1	-1		1	-2	1

Usage

```
imgSharpen (imgdata, mask)
```

Arguments

imgdata	The image
mask	The matrix to be used in the convolution. Must be one of 1, 2, 3 (default=1)

Value

return an imagedata object

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgSharpen(x, 2)

## End(Not run)
```

imgShenCastan

*Shen-Castan Edge Detection Method***Description**

This function does edge detection using the Shen-Castan algorithm.

Usage

```
imgShenCastan(imgdata, smooth_factor=0.9, thin_factor=2, adapt_window=7, thresh_ratio=0.8, do_hystere
```

Arguments

```
imgdata      The image
smooth_factor The smooth factor
thin_factor  The thinning factor
adapt_window The size of the window for adaptive gradient
thresh_ratio The percentage of pixels to be above high threshold
do_hysteresis If true, do hysteresis
```

Value

return an imagedata object

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgShenCastan(x)

## End(Not run)
```

imgSobel

Sobel Edge Detection Method

Description

This function enhances image's edges by convoluting with the Sobel method matrices:

	H_r				H_c		
1	0	-1		-1	-2	-1	
2	0	-2		0	0	0	
1	0	-1		1	2	1	

Usage

```
imgSobel(imgdata)
```

Arguments

```
imgdata      The image
```

Value

return an imagedata object

Examples

```
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgSobel(x)
```

imgSplineRotate	<i>Rotate an image</i>
-----------------	------------------------

Description

This function rotates an image using b-spline interpolation and returns a new image.

Usage

```
imgSplineRotate(imgdata, angle)
```

Arguments

imgdata	The image
angle	The clockwise deg angle to rotate

Value

return an imagedata object

See Also

[imgRotate](#) [imgNearestNeighborRotate](#) [imgBilinearRotate](#) [imgCubicRotate](#) [imgRotate90Clockwise](#)
[imgRotate90CounterClockwise](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgSplineRotate(x, 45)

## End(Not run)
```

imgSplineScale	<i>Scale an image</i>
----------------	-----------------------

Description

This function scales an image using b-spline interpolation and returns a new image.

Usage

```
imgSplineScale(imgdata, x_scale, y_scale)
```

Arguments

imgdata	The image
x_scale	The horizontal scale factor
y_scale	The vertical scale factor

Value

return an imagedata object

Note

The scale factors are expected to be greater than 1. To reduce an image use the minification functions instead.

See Also

[imgScale](#) [imgNearestNeighborScale](#) [imgBilinearScale](#) [imgCubicScale](#) [imgMedianShrink](#) [imgAverageShrink](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgSplineScale(x, 1.5, 1.5)  
  
## End(Not run)
```

imgStdBinaryClosing *Fixed mask binary closing*

Description

This function applies a Binary Closing with a 0-squared mask, with given dimension

Usage

```
imgStdBinaryClosing(imgdata, dim)
```

Arguments

imgdata	The image
dim	mask's dimension (default = 3)

Value

return an imagedata object

Note

This function accepts binary images only and will treat gray scale ones as binary images.

See Also

[imgStdBinaryErosion](#) [imgStdBinaryDilation](#) [imgBinaryClosing](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgStdBinaryClosing(x, 4)  
  
## End(Not run)
```

imgStdBinaryDilation *Fixed mask binary dilation*

Description

This function makes a dilation of a binary image with a 0-squared mask, with given dimension.

Usage

```
imgStdBinaryDilation(imgdata, dim)
```

Arguments

imgdata	The image
dim	mask's dimension (default = 3)

Value

return an imagedata object

Note

This function accepts binary images only and will treat gray scale ones as binary images.

See Also

[imgStdBinaryDilation](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgStdBinaryDilation(x, 4)  
  
## End(Not run)
```

imgStdBinaryErosion *Fixed mask binary erosion*

Description

This function makes an erosion of a binary image with a 0-squared mask, with given dimension.

Usage

```
imgStdBinaryErosion(imgdata, dim)
```

Arguments

imgdata	The image
dim	mask's dimension (default = 3)

Value

return an imagedata object

Note

This function accepts binary images only and will treat gray scale ones as binary images.

See Also[imgStdBinaryErosion](#)**Examples**

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgStdBinaryErosion(x, 4)

## End(Not run)
```

imgStdBinaryOpening *Fixed mask binary opening*

Description

This function applies a Binary Opening with a 0-squared mask, with given dimension

Usage

```
imgStdBinaryOpening(imgdata, dim)
```

Arguments

imgdata	The image
dim	mask's dimension (default = 3)

Value

return an imagedata object

Note

This function accepts binary images only and will treat gray scale ones as binary images.

See Also[imgStdBinaryErosion](#) [imgStdBinaryDilation](#) [imgBinaryOpening](#)**Examples**

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgStdBinaryOpening(x, 4)

## End(Not run)
```

imgStdBlur	<i>Blurs an image</i>
------------	-----------------------

Description

This function blurs an image by convoluting with a average square matrix

Usage

```
imgStdBlur(imgdata, dim)
```

Arguments

imgdata	The image
dim	Square matrix dimension (optional, default = 5)

Value

return an imagedata object

See Also

[imgBlur](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgStdBlur(x, 3)  
  
## End(Not run)
```

imgStdNDilationErosion	<i>Fixed mask NDilationErosion</i>
------------------------	------------------------------------

Description

This function applies dilation n times and then erosion n times, with a 0-squared matrix with a given dimension.

Usage

```
imgStdNDilationErosion(imgdata, n, dim=3)
```

Arguments

imgdata	The image
n	Times to apply each operation
dim	mask's dimension (default = 3)

Value

return an imagedata object

Note

This function accepts binary images only and will treat gray scale ones as binary images.

See Also

[imgNDilationErosion](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgStdNDilationErosion(x, 4, 5)  
  
## End(Not run)
```

imgStdNErosionDilation

Fixed mask NErosionDilation

Description

This function applies erosion n times and then dilation n times, with a 0-squared matrix with a given dimension.

Usage

```
imgStdNErosionDilation(imgdata, n, dim=3)
```

Arguments

imgdata	The image
n	Times to apply each operation
dim	mask's dimension (default = 3)

Value

return an imagedata object

Note

This function accepts binary images only and will treat gray scale ones as binary images.

See Also

[imgNErosionDilation](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgStdNErosionDilation(x, 4, 5)

## End(Not run)
```

imgThreshold	<i>Threshold an image</i>
--------------	---------------------------

Description

This function thresholds an image using a given filter.

Usage

```
imgThreshold(imgdata, thr_value)
```

Arguments

imgdata	The image
thr_value	Filter value for thresholding

Value

return an imagedata object

See Also

[r_threshold](#)

Examples

```
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgThreshold(x, 80)
```

imgTranslate *Translate an image block*

Description

This function translates an image block and returns a new image.

Usage

```
imgTranslate(imgdata, x_start, y_start, x_end, y_end, t_width, t_height)
```

Arguments

imgdata	The image
x_start	Upper left x coordinate of source block
y_start	Upper left y coordinate of source block
x_end	Upper left x coordinate of destination block
y_end	Upper left y coordinate of destination block
t_width	Width of the block to move
t_height	Height of the block to move

Value

return an imagedata object

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- imgTranslate(x, 100, 100, 200, 200, 50, 50)

## End(Not run)
```

imgUnsharpen *Unsharpens an image with selected mask*

Description

This function unsharpens an image by convoluting with one of the following matrices:

1		2		3	1
0 -1 0		-1 -1 -1		1 -2 1	
-1 5 -1		-1 9 -1		-2 5 -2	
0 -1 0		-1 -1 -1		1 -2 1	

Performs a difference between original image and sharpen convolved image with the specified mask

Usage

```
imgUnsharpen (imgdata, mask)
```

Arguments

imgdata	The image
mask	The matrix to be used in the convolution. Must be one of 1, 2, 3 (default=1)

Value

return an imagedata object

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgUnsharpen(x, 2)  
  
## End(Not run)
```

imgVerticalMirroring *Vertical mirror an image*

Description

This function flips an image about the x axis.

Usage

```
imgVerticalMirroring(imgdata)
```

Arguments

imgdata	The image
---------	-----------

Value

return an imagedata object

See Also

[imgHorizontalMirroring](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgVerticalMirroring(x)  
  
## End(Not run)
```

imgXOR

Xor two images

Description

This function does a logic XOR between two images and returns a new image.

Usage

```
imgXOR(imgdata1, imgdata2)
```

Arguments

imgdata1	The first image
imgdata2	The second image

Value

return an imagedata object

See Also

[imgOR](#) [imgAND](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- imgXOR(x, x)  
  
## End(Not run)
```

logo	<i>R logo imagedata</i>
------	-------------------------

Description

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

Usage

```
data(logo)
```

Format

```
imagedata
```

Examples

```
## Not run:  
data(logo)  
plot(logo)  
  
## End(Not run)
```

plot.imagedata	<i>Plotting an imagedata object</i>
----------------	-------------------------------------

Description

This function outputs an imagedata object as an image.

Usage

```
plot.imagedata(x, ...)
```

Arguments

x	The image
...	Plotting options

See Also

[imagedata](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
plot(x)  
  
## End(Not run)
```

print.imagedata	<i>Print information on a given imagedata object</i>
-----------------	--

Description

This function outputs information on a given imagedata object.

Usage

```
print.imagedata(x, ...)
```

Arguments

x	The image
...	Ignored

See Also

[imagedata](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
print(x)  
  
## End(Not run)
```

readJpeg	<i>Read jpeg file</i>
----------	-----------------------

Description

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

Usage

```
readJpeg(filename)
```

Arguments

filename filename of JPEG image

Value

return an imagedata object

See Also

[imagedata](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
plot(x)  
  
## End(Not run)
```

readTiff	<i>Read tiff file</i>
----------	-----------------------

Description

This function reads a tiff image file and return an imagedata object.

Usage

```
readTiff(filename)
```

Arguments

filename filename of TIFF image

Value

return an imagedata object

See Also

[imagedata](#)

Examples

```
## Not run:  
x <- readTiff(system.file("samples", "violet.tif", package="biOps"))  
plot(x)  
  
## End(Not run)
```

r_dec_contrast	<i>Decrease contrast</i>
----------------	--------------------------

Description

This function decreases an image contrast, leaving each pixel value between given values.

Usage

```
r_dec_contrast(imgdata, min_desired, max_desired)
```

Arguments

imgdata	The image
min_desired	The min value
max_desired	The max value

Value

return an imagedata object

Note

This is the R implementation of imgDecreaseContrast.

See Also

[imgDecreaseContrast](#) [imgIncreaseContrast](#) [r_inc_contrast](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- r_dec_contrast(x, 60, 200)  
  
## End(Not run)
```

r_dec_intensity	<i>Decrease intensity</i>
-----------------	---------------------------

Description

This function decreases an image intensity by a given factor.

Usage

```
r_dec_intensity(imgdata, percentage)
```

Arguments

imgdata	The image
percentage	A non negative value representing the intensity percentage to be decreased. 1 stands for 100% (eg. 0.5 = 50%).

Value

return an imagedata object

Note

This is the R implementation of imgDecreaseIntensity.

See Also

[imgDecreaseIntensity](#) [imgIncreaseIntensity](#) [r_inc_intensity](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- r_dec_intensity(x, 0.3)  
  
## End(Not run)
```

r_gamma	<i>Gamma correct an image</i>
---------	-------------------------------

Description

This function applies gamma operation to a given image. Each pixel value is taken to the inverse of gamma_value-th exponent

Usage

```
r_gamma(imgdata, gamma_value)
```

Arguments

imgdata	The image
gamma_value	A non negative value representing operation gamma value

Value

return an imagedata object

Note

This is the R implementation of imgGamma.

See Also

[imgGamma](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- r_gamma(x, 1.3)  
  
## End(Not run)
```

r_imgAdd	<i>Add two images</i>
----------	-----------------------

Description

This function adds two images and returns a new image.

Usage

```
r_imgAdd(imgdata1, imgdata2)
```

Arguments

imgdata1	The first image
imgdata2	The second image

Value

return an imagedata object

Note

This is the R implementation of imgAdd.

See Also

[imgAdd](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- r_imgAdd(x, x)  
  
## End(Not run)
```

r_imgAverage	<i>Average images</i>
--------------	-----------------------

Description

This function calculates the average of the given images and returns a new image.

Usage

```
r_imgAverage(imgdata_list)
```

Arguments

imgdata_list An image list

Value

return an imagedata object

Note

This is the R implementation of imgAverage.

See Also

[imgAverage](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- r_imgAverage(list(x, x))

## End(Not run)
```

r_imgDiffer	<i>Subtract two images</i>
-------------	----------------------------

Description

This function subtracts two images and returns a new image, `imgdata1 - imgdata2`.

Usage

```
r_imgDiffer(imgdata1, imgdata2)
```

Arguments

imgdata1 The first image
imgdata2 The second image

Value

return an imagedata object

Note

This is the R implementation of imgDiffer.

See Also[imgDiffer](#)**Examples**

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- r_imgDiffer(x, x)

## End(Not run)
```

r_imgMaximum	<i>Images maximum</i>
--------------	-----------------------

Description

This function calculates the maximum of the given images and returns a new image.

Usage

```
r_imgMaximum(imgdata_list)
```

Arguments

imgdata_list An image list

Value

return an imagedata object

Note

This is the R implementation of imgAverage.

See Also[imgMaximum](#)**Examples**

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- r_imgMaximum(list(x, x))

## End(Not run)
```

r_inc_contrast	<i>Increase contrast</i>
----------------	--------------------------

Description

This function increases an image contrast, augmenting pixel values differences between given limits (in a linear fashion).

Usage

```
r_inc_contrast(imgdata, min_limit, max_limit)
```

Arguments

imgdata	The image
min_limit	The minimum limit to apply lineal modification
max_limit	The maximum limit to apply lineal modification

Value

return an imagedata object

Note

This is the R implementation of `imgIncreaseContrast`.

See Also

[imgIncreaseContrast](#) [imgDecreaseContrast](#) [r_dec_contrast](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- r_inc_contrast(x, 60, 200)  
  
## End(Not run)
```

r_inc_intensity	<i>Increase intensity</i>
-----------------	---------------------------

Description

This function increases an image intensity by a given factor.

Usage

```
r_inc_intensity(imgdata, percentage)
```

Arguments

imgdata	The image
percentage	A non negative value representing the intensity percentage to be increased. 1 stands for 100% (eg. 0.5 = 50%).

Value

return an imagedata object

Note

This is the R implementation of imgIncreaseIntensity.

See Also

[imgIncreaseIntensity](#) [imgDecreaseIntensity](#) [r_dec_intensity](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- r_inc_intensity(x, 0.3)  
  
## End(Not run)
```

r_look_up_table	<i>Transforms an image by a given look-up table</i>
-----------------	---

Description

This function applies a transformation to an image using a given look-up table.

Usage

```
r_look_up_table(imgdata, table)
```

Arguments

imgdata	The image
table	Look up table which determines the image operation to be applied

Value

return an imagedata object

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
lut <- seq(255, 0, by=-1)  
y <- r_threshold(x, lut)  
  
## End(Not run)
```

r_negative	<i>Negate an image</i>
------------	------------------------

Description

This function negates an image.

Usage

```
r_negative(imgdata)
```

Arguments

imgdata	The image
---------	-----------

Value

return an imagedata object

Note

This is the R implementation of `imgNegative`.

See Also

[imgNegative](#) [r_negative_lut](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- r_negative(x)

## End(Not run)
```

r_negative_lut	<i>Negate an image</i>
----------------	------------------------

Description

This function negates an image.

Usage

```
r_negative_lut(imgdata)
```

Arguments

imgdata The image

Value

return an imagedata object

Note

This is the R implementation of `imgNegative` using look up tables.

See Also

[imgNegative](#) [r_negative](#)

Examples

```
## Not run:
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
y <- r_negative_lut(x)

## End(Not run)
```

r_threshold *Threshold an image*

Description

This function thresholds an image using a given filter.

Usage

```
r_threshold(imgdata, thr_value)
```

Arguments

imgdata	The image
thr_value	Filter value for thresholding

Value

return an imagedata object

Note

This is the R implementation of imgThreshold.

See Also

[imgThreshold](#)

Examples

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
y <- r_threshold(x, 80)  
  
## End(Not run)
```

writeJpeg *Write jpeg file*

Description

This function writes an imagedata object into a jpeg image file.

Usage

```
writeJpeg(filename, imgdata)
```

Arguments

filename	filename of JPEG image
imgdata	imagedata to write

See Also[readJpeg](#)**Examples**

```
## Not run:  
x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))  
writeJpeg("new_image.jpg", x)  
  
## End(Not run)
```

writeTiff	<i>Write tiff file</i>
-----------	------------------------

Description

This function writes an imagedata object into a tiff image file.

Usage

```
writeTiff(filename, imgdata)
```

Arguments

filename	filename of TIFF image
imgdata	imagedata to write

See Also[readTiff](#)**Examples**

```
## Not run:  
x <- readTiff(system.file("samples", "violet.tif", package="biOps"))  
writeTiff("new_image.tif", x)  
  
## End(Not run)
```

Index

*Topic **IO**

- readJpeg, 84
- readTiff, 85
- writeJpeg, 96
- writeTiff, 97

*Topic **datasets**

- logo, 83

*Topic **logic**

- imgAND, 10
- imgNegative, 58
- imgOR, 60
- imgXOR, 82
- r_negative, 94
- r_negative_lut, 95

*Topic **math**

- imgAdd, 9
- imgAverage, 11
- imgAverageShrink, 11
- imgBilinearRotate, 12
- imgBilinearScale, 13
- imgBinaryClosing, 14
- imgBinaryDilation, 15
- imgBinaryErosion, 15
- imgBinaryOpening, 16
- imgBlockMedianFilter, 17
- imgBlur, 18
- imgBoost, 19
- imgCanny, 20
- imgConvolve, 21
- imgCrop, 21
- imgCubicRotate, 22
- imgCubicScale, 23
- imgDecreaseContrast, 24
- imgDecreaseIntensity, 24
- imgDiffer, 25
- imgDifferenceEdgeDetection, 26
- imgDivide, 27
- imgEKMeans, 27
- imgFFT, 28

- imgFFTBandPass, 29
- imgFFTBandStop, 30
- imgFFTConvolve, 30
- imgFFTHighPass, 31
- imgFFTInv, 32
- imgFFTiShift, 33
- imgFFTLowPass, 33
- imgFFTPhase, 34
- imgFFTShift, 35
- imgFFTSpectrum, 35
- imgFreiChen, 36
- imgGamma, 37
- imgGaussianNoise, 37
- imgGrayScaleClosing, 39
- imgGrayScaleDilation, 40
- imgGrayScaleErosion, 40
- imgGrayScaleOpening, 41
- imgHighPassFilter, 43
- imgHomogeneityEdgeDetection, 45
- imgHorizontalMirroring, 46
- imgIncreaseContrast, 46
- imgIncreaseIntensity, 47
- imgIsoData, 48
- imgKDKMeans, 49
- imgKirsch, 49
- imgKMeans, 50
- imgMarrHildreth, 51
- imgMaximum, 51
- imgMaximumFilter, 52
- imgMedianShrink, 53
- imgMinimumFilter, 54
- imgMultiply, 54
- imgNDilationErosion, 55
- imgNearestNeighborRotate, 56
- imgNearestNeighborScale, 57
- imgNErosionDilation, 58
- imgPadding, 61
- imgPrewitt, 61
- imgPrewittCompassGradient, 62

- imgRoberts, 64
- imgRobinson3Level, 65
- imgRobinson5Level, 65
- imgRotate, 66
- imgRotate90Clockwise, 67
- imgRotate90CounterClockwise, 68
- imgSaltPepperNoise, 68
- imgScale, 69
- imgSharpen, 70
- imgShenCastan, 70
- imgSobel, 71
- imgSplineRotate, 72
- imgSplineScale, 73
- imgStdBinaryClosing, 74
- imgStdBinaryDilation, 74
- imgStdBinaryErosion, 75
- imgStdBinaryOpening, 76
- imgStdBlur, 77
- imgStdNDilationErosion, 77
- imgStdNErosionDilation, 78
- imgThreshold, 79
- imgTranslate, 80
- imgUnsharpen, 80
- imgVerticalMirroring, 81
- r_dec_contrast, 86
- r_dec_intensity, 87
- r_gamma, 88
- r_imgAdd, 89
- r_imgAverage, 89
- r_imgDiffer, 90
- r_imgMaximum, 91
- r_inc_contrast, 92
- r_inc_intensity, 93
- r_look_up_table, 94
- r_threshold, 96
- *Topic **misc**
 - imagedata, 7
 - imageType, 9
 - imgBlueBand, 18
 - imgGetRGBFromBands, 38
 - imgGreenBand, 42
 - imgHistogram, 44
 - imgNormalize, 59
 - imgRedBand, 63
 - imgRGB2Grey, 63
 - plot.imagedata, 83
 - print.imagedata, 84
- *Topic **package**
 - biOps-package, 4
 - biOps (biOps-package), 4
 - biOps-package, 4
 - hist, 45
 - imagedata, 7, 83–85
 - imageType, 9
 - imgAdd, 9, 89
 - imgAND, 10, 60, 82
 - imgAverage, 11, 90
 - imgAverageShrink, 11, 13, 23, 53, 57, 69, 73
 - imgBilinearRotate, 12, 22, 56, 66, 72
 - imgBilinearScale, 12, 13, 23, 53, 57, 69, 73
 - imgBinaryClosing, 14, 74
 - imgBinaryDilation, 14, 15, 17, 39, 42, 55, 59
 - imgBinaryErosion, 14, 15, 17, 39, 42, 55, 59
 - imgBinaryOpening, 16, 76
 - imgBlockMedianFilter, 17
 - imgBlueBand, 18
 - imgBlur, 18, 77
 - imgBoost, 19
 - imgCanny, 20
 - imgConvolve, 21
 - imgCrop, 21
 - imgCubicRotate, 12, 22, 56, 66, 72
 - imgCubicScale, 12, 13, 23, 53, 57, 69, 73
 - imgDecreaseContrast, 24, 47, 86, 92
 - imgDecreaseIntensity, 24, 47, 87, 93
 - imgDiffer, 25, 91
 - imgDifferenceEdgeDetection, 26
 - imgDivide, 27
 - imgEKMeans, 27, 48–50
 - imgFFT, 28, 29–36
 - imgFFTBandPass, 29, 30, 31, 34
 - imgFFTBandStop, 29, 30, 31, 34
 - imgFFTConvolve, 30
 - imgFFTHighPass, 29, 30, 31, 34
 - imgFFTIInv, 28–31, 32, 33–36
 - imgFFTiShift, 28, 32, 33, 35
 - imgFFTLowPass, 29–31, 33
 - imgFFTPhase, 34, 36
 - imgFFTShift, 28, 32, 33, 35
 - imgFFTSpectrum, 34, 35
 - imgFreiChen, 36
 - imgGamma, 37, 88
 - imgGaussianNoise, 37, 69
 - imgGetRGBFromBands, 38

- imgGrayScaleClosing, 39
- imgGrayScaleDilation, 39, 40, 42
- imgGrayScaleErosion, 39, 40, 42
- imgGrayScaleOpening, 41
- imgGreenBand, 42
- imgHighPassFilter, 19, 43
- imgHistogram, 44
- imgHomogeneityEdgeDetection, 26, 45, 45
- imgHorizontalMirroring, 46, 81
- imgIncreaseContrast, 24, 46, 86, 92
- imgIncreaseIntensity, 25, 47, 87, 93
- imgIsoData, 28, 48, 49, 50
- imgKDKMeans, 28, 48, 49, 50
- imgKirsch, 49
- imgKMeans, 28, 48, 49, 50
- imgMarrHildreth, 51
- imgMaximum, 51, 91
- imgMaximumFilter, 52
- imgMedianShrink, 12, 13, 23, 53, 57, 69, 73
- imgMinimumFilter, 54
- imgMultiply, 54
- imgNDilationErosion, 55, 78
- imgNearestNeighborRotate, 12, 22, 56, 66, 72
- imgNearestNeighborScale, 12, 13, 23, 53, 57, 69, 73
- imgNegative, 58, 95
- imgNErosionDilation, 58, 79
- imgNormalize, 59
- imgOR, 10, 60, 82
- imgPadding, 61
- imgPrewitt, 61
- imgPrewittCompassGradient, 62
- imgRedBand, 63
- imgRGB2Grey, 63
- imgRoberts, 64
- imgRobinson3Level, 65
- imgRobinson5Level, 65
- imgRotate, 12, 22, 56, 66, 72
- imgRotate90Clockwise, 12, 22, 56, 66, 67, 68, 72
- imgRotate90CounterClockwise, 12, 22, 56, 66, 67, 68, 72
- imgSaltPepperNoise, 38, 68
- imgScale, 13, 23, 57, 69, 73
- imgSharpen, 70
- imgShenCastan, 70
- imgSobel, 71
- imgSplineRotate, 12, 22, 56, 66, 72
- imgSplineScale, 13, 23, 57, 69, 73
- imgStdBinaryClosing, 74
- imgStdBinaryDilation, 74, 74–76
- imgStdBinaryErosion, 74, 75, 76
- imgStdBinaryOpening, 76
- imgStdBlur, 19, 77
- imgStdNDilationErosion, 77
- imgStdNErosionDilation, 78
- imgThreshold, 79, 96
- imgTranslate, 80
- imgUnsharpen, 80
- imgVerticalMirroring, 46, 81
- imgXOR, 10, 60, 82
- logo, 83
- plot.imagedata, 8, 83
- print.imagedata, 8, 84
- r_dec_contrast, 24, 47, 86, 92
- r_dec_intensity, 25, 47, 87, 93
- r_gamma, 37, 88
- r_imgAdd, 89
- r_imgAverage, 89
- r_imgDiffer, 90
- r_imgMaximum, 91
- r_inc_contrast, 24, 47, 86, 92
- r_inc_intensity, 25, 47, 87, 93
- r_look_up_table, 94
- r_negative, 58, 94, 95
- r_negative_lut, 58, 95, 95
- r_threshold, 79, 96
- readJpeg, 84, 97
- readTiff, 85, 97
- writeJpeg, 96
- writeTiff, 97