EBImage

April 19, 2009

`EBImage-deprecated` *EBImage deprecated functions*

Description

The functions listed below are deprecated and their new variants should be used instead. The functions are likely to be defunct with the next Bioconductor release, 2.2. The functions are deprecated due to the naming issues (the use of a dot) and new variants carry the same arguments as the deprecated ones.

Substitutions for the deprecated functions

```
'choose.image' => chooseImage
'edge.features' => edgeFeatures
'edge.profile' => edgeProfile
'haralick.features' => haralickFeatures
'haralick.matrix' => haralickMatrix
'hull.features' => hullFeatures
'read.image' => readImage
'write.image' => writeImage
'zernike.moments' => zernikeMoments
```

`EBImage-package` *Package overview*

Description

`EBImage` is the image processing and analysis package for R. Its primary goal is to enable automated analysis of large sets of images such as those obtained in high throughput automated microscopy.

The package uses `ImageMagick` library for image I/O operations and for many image processing routines. Algorithms for image analysis are implemented natively including algorithms for feature extraction.

Image data are stored in objects of the class `Image`, which is derived as an S4 class from `array` and inherits all of its properties and methods.

The project page is `http://www.ebi.ac.uk/~osklyar/EBImage`

Package content

Classes

- Class 'Image', its accessor method
- Class 'IndexedImage'
- Common generic methods for class 'Image'
- 'Image' object creation, copying and assertion
- Image read/write operations

Image processing

- Enhancing images and colors
- Image color manipulation
- Generate a tiled image from a stack
- Image and color channel thresholding
- Image transformation, rotation, resize etc.
- Noise removal, blurring and smoothing of images

Image analysis

- Distance map transform of binary images
- Morphological trasnsformations of binary images
- Segmentation and edge detection
- Voronoi-based segmentation on image manifolds
- Watershed transformation and watershed-based object detection

Feature extraction

- Extraction of Haralick texture features and co-occurance matrices (GLCM)
- Extraction of edge profiles and edge features
- Extraction of hull features
- Extraction of Zernike moments
- Extraction of image moments and moment invariants
- Combined feature extraction for objects in indexed images
- Object removal in indexed images
- Marking detected objects in indexed images
- Generate a stack of images for detected objects
- Matching objects in two indexed images

Tools

- Color and image color mode conversions
- Drawing primitives on images, annotation
- Interactive image display

Authors

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, Copyright 2005-2007

Wolfgang Huber, ⟨huber@ebi.ac.uk⟩

Mike Smith, ⟨msmith@ebi.ac.uk⟩

European Bioinformatics Institute
European Molecular Biology Laboratory
Wellcome Trust Genome Campus
Hinxton
Cambridge CB10 1SD
UK

---

image,Image-method *Generic methods for objects of class Image*

Description

Image supports all (or almost all) operations that are defined for array. However, the default behavior of some generics was redefined in the package to better suit the class. Those are:

Generic methods

show(object) Prints a summary of image and properties (as described in 'Accessor methods' section). Prints also an excerpt of data keeping it to the minimum just to have a visual control over the atomic data type. Use imageData(x) to print or assign the full dataset. If image was put through the object detection algorithm a summary of detected objects will be printed instead of data.

as.array(x) Removed to avoid incompatibilities with R < 2.5. Use imageData instead.

as.matrix(x, ...) Removed to avoid incompatibilities with R < 2.5. Use as.matrix(imageData(x)) instead.

hist(x, ...) A histogramme method.

median(x, na.rm=FALSE) stats:median redefinition for images (bug workaround: without it median takes forever to execute without no apparent reason)

image(x, ...) Draws image data using graphics::image keeping image aspect ratio.

[Redefined versions (multiple signatures) of the array subsetting operator. These return objects of class Image without dropping dimensions.

Arith Redefined arithmetic operators acting on any combination of Image with another Image or array will return an object of class Image.

as.Image)(x, ...) S4 method for the signature x=array. Is equivalent to Image(x) when x is array.

Other important methods that directly applicable due to inheritance from the array class, just to stress that they can and should be used:

dim(x) and dim(x) <- value Gets and sets image dimensions.

fft(z, inverse = FALSE) Performs the Fast Fourier Transformation on an Image.

sqrt(x) etc. Miscellaneous mathematical functions.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2006-2007

See Also

Image, Image

---

channel *Color and image color mode conversions*

Description

channel allows to convert colors of images and color values given in vectors, arrays or matrices from one mode to another. The supported source and target modes are TrueColor (data of type: integer) and Grayscale (numeric) and X11 strings (character).

The mode of the argument is determined automatically from its type.

Usage

```
## S4 methods for signatures 'x=Image,mode=character'
## and 'x=ANY,mode=character'
channel(x, mode, ...)

rgbImage(r, g, b)
channelMix(r, g, b)
```

Arguments

x	Either an object of Image or a vector of numeric, integer or character.
mode	A character value specifying the target mode for conversion. See 'Conversion Modes' below.
...	Reserved.
r,g,b	Objects of class Image.

Details

Conversion modes:

rgb The result is a true color representation of the input. Its class is or includes integer, i.e. an integer array or a TrueColor integer-based image.

gray The result is a gray scale representation of the input. Its class is or includes numeric. Synonym: grey

red, green, blue The result is a gray scale representation of the input from only the red, the green or the blue channel was extracted, correspondingly. The result's class is numeric-based. As synonyms single letters r, g, b can be used correspondingly.

asred, asgreen, asblue The result is a true color with all the data in one channel only, the red, green or blue, correspondingly. The result's class is integer-based. As synonyms 2r, 2g, 2b can be used.

x11 The result is a character, which elements represent the input colors as X11 strings, i.e. RGB hex codes starting with a sharp sign.

Conversion modes are case insensitive!

If the input is a character vector, its values may contain color names instead of color hex codes, like red, purple, white etc. However, the result of the X11 mode will always contain hex codes.

channelMix uses the following formula to calculate the mix: mix = sqrt(r^2+g^+b^2). Any of the values can be missing.

Value

For an argument of Image the function returns a new object of Image in all modes except 'X11'. In case of 'X11' it returns a character vector of the same length as the image.

For vectors the function returns a vector of the same size in the target mode.

rgbImage generated a TrueColor image using r, g and b as values for red, green and blue channels correspondingly.

channelMix generates a grayscale image as if the image was converted from a TrueColor one that was composed using r, g and b.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩

See Also

Image, Image, Image

Examples

```
cl <- channel("pink", "rgb")
r <- channel( channel(cl, "red"), "asred")
g <- channel( channel("pink", "green"), "asgreen")
b <- channel( channel(cl, "blue"), "asblue")
rgb <- r + g + b
print( rgb == cl )
clg <- channel(cl, "gray")
print( channel(cl, "x11"))
```

```
## further example in 'paintObjects'
```

`Image-class` *Defintion of class 'Image', its accessor methods*

Description

The class `Image` enables the storage and manipulation of image data for grayscale and true color (RGB) images in `R`. The class is based on (derived from) `array` and inherits all its properties and methods, e.g. mathematical operations, subsetting, histograms etc. Image processing and analysis routines are defined as S4 methods for objects of the class `Image`.

Class Definition

The class is defined as follows:

```
## S4 class definition

setClass( "Image",
  representation (
    colormode    = "integer",
    filename     = "character",
    compression  = "character",
    resolution   = "numeric",
    features     = "list"
  ),
  contains = "array"
)
```

Additionally, two constants are defined and exported that can be used to specify or test image data type:

```
  Grayscale = as.integer(0)
  TrueColor = as.integer(1)
```

Creating objects

Objects can be created using `new("Image")` supplying slot data as necessary. Additionally, wrapper functions are defined to simplify the construction of `Image` objects in different situations: the default constructor, `Image`, the `readImage` constructor that creates an object from reading image files and the `copy` constructor.

Accessor methods

Accessor methods are class methods that are defined to retrieve data values from objects of a given class or to assigne those. These should be used instead of directly referring to the slots. Unfortunately, R does not have a concept of private or protected class members: all slots have "public" access in terms of other object-oriented languages. Therefore, data encapsulation in sense of hiding the particular structure of the class is impossible. At the same time, it is still recommended to use accessor functions instead of directly accessing slots. One can also assume that any slot that does not have an accessor method should not be accessed directly by the user.

The following accessor methods are defined as S4 methods for class `Image`:

colorMode(x), colorMode(x) <- value Gets and sets image color mode. value is one of the constants above. Default: Grayscale.

compression(x), compression(x) <- value Gets and sets compression algorithm used when image saved in a format supporting compression. For exapmple, TIFF images support ZIP and LZW lossless compressions, whereas JPEG uses JPEG compression by default. Possible values of type character are 'NONE', 'LZW', 'ZIP', 'JPEG', 'BZIP' and 'GROUP4'. Some of these may be unavailable on your system if ImageMagick was compiles without their support. Default: 'LZW'.

features(x) Is a read only method that returns a list or matrices with descriptors of detected objects. If object detection was executed on the image, an empty list is returned. Otherwise, the number of elements matches the number of frames in the image. Default: list().

fileName(x), fileName(x) <- value Gets and sets the file name. When reading images this value is updated, when writing it can be used as default if no alternative is specified. When reading a stack, the first file name in the list of stack images will be assigned. Default: 'no-name'.

imageData(x), imageData(x) <- value Gets and sets image data. Image data is a 3D array with the last dimension matching the number of 2D images in a stack. The atomic type of the array is defined by colorMode().

resolution(x), resolution(x) <- value Gets and sets values of image resoultion: a numeric vector of two values of resolution in x and y dimensions. Generally resolution is understood in pixel-per-inch (ppi) or dots-per-inch (dpi), however some microscopes can save these values in images in different units. It is the responsibility of the user to identify the unit and keep the units comparable between different images in the same project if required: the package does not take care of this. The default values are equal for both directions, 2.5e+6 dpi.

Details

Image data are stored as 3D arrays with first two dimensions specifying frame size and third the number of frames. A single 2D image has the number of frames 1. In subsetting, the value of drop parameter is set to FALSE, thus image dimensionality is preserved at 3.

Image is derived from array, therefore, the data is stored in arrays. Image can store either grayscale or true color data at the moment. Grayscale data is stored in numeric arrays, whereas true color data is stored in integer arrays. Correspondingly, the data is created with array(as.numeric(val),dim(val)) for grayscale and array(as.integer(val),dim(val)) for true color images. Grayscale values are assumed to be in the range [0,1], although this is not a requirement in sense of data storage. Although, many image processing functions will assume data in this range or will generate invalid results for the data out of this range.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2005-2007

See Also

Image, Image, Image, Image, Image, Image,

Examples

```
## New Grayscale image of a black-to-white vertical gradient
w <- 120
```

```
a <- Image((0:(w^2))/w^2, c(w,w))
if ( interactive() ) display(a)
print(a)

## Converts image to TrueColor
b <- a
colorMode(b) <- TrueColor
print(b)

## Fills values with intensities over 0.5 (50%) with red
b[ a > 0.5 ] = as.integer(255)
if ( interactive() ) display(b)
```

IndexedImage-class *Defintion of class 'IndexedImage'*

Description

The class IndexedImage is used to store the results of functions that index image into separate objects using integer 1-based indexing. Such an image is essentially the same as Image in the gray scale mode. The class was defined to ensure correct types in calls to other object detection and processing routines.

Class Definition

```
## S4 class definition
setClass( "IndexedImage", contains="Image" )
```

Creating objects

Objects of this class are not supposed to be created directly, rather as the result of calls to watershed or propagate, or other functions that index objects in images.

Details

No accessor methods are defined for this class, however all of the parents' ones are available, i.e. those of Image and array.

One coersion routine is defined in addition, which simply sets the class attribute to Image:

```
## S4 method for signature 'x=IndexedImage'
as.Image(x, ...)
```

When used with IndexedImage's display by default normalizes the image.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2007

See Also

Image, Image, Image

Examples

```
## load images
f <- paste( system.file(package="EBImage"), "images/Gene1_G.tif", sep="/" )
ii = readImage(f)
## normalize images
ii = normalize(ii, separate=TRUE)
## segment
mask = thresh(ii, 25, 25, 0.02)
## refine segmentation with morphology filters
mk3 = morphKern(3)
mk5 = morphKern(5)
mask = dilate(erode(closing(mask, mk5), mk3), mk5)
## index objects with 'watershed'
io = watershed( distmap(mask), 1.5, 1)

class(io)
```

`Image` *Image creation, copying and assertion*

Description

Functions to create, copy and assert images.

Usage

```
Image(data=array(0,c(0,0,1)), dim=base::dim(data), colormode, ...)

is.Image(x)
stopIfNotImage(x)

## S4 methods for signature 'x=Image'
copy(x, ...)
header(x, ...)

## S4 methods for signatures 'x=Image,y=Image'
## and 'x=list,y=missing'
combine(x, y, ...)

## S4 methods for signatures 'x=Image,y=Image'
## and 'x=Image,y=missing'
assert(x, y, strict=FALSE, ...)
```

Arguments

`x, y`	Objects of class `Image`. There is also a `combine` method for lists of equally sized `Image` objects, in that case `y` should be missing.
`data`	Data to fill the image, typically an array, but can be any object for which `as.numeric` or `as.integer` is defined.
`dim`	A numeric vector of image dimensions with length of 2 or 3. If its length is 2, the third dimension is set to 1.

`colormode`	An integer value for the image data color mode. It is recommended to use the predefined symbols `TrueColor` or `Grayscale`.
`strict`	A logical scalar. If `TRUE`, the size of all three dimensions of two images will be compared, if `FALSE`, the function will compare only the first two dimensions (i.e. stacks can have different size in z-direction, but x- and y-size should be the same).
`...`	With `Image`, further arguments to `new`; with `combine`, further images to be combined.

Details

`Image` This is a wrapper around `new`, for convenience.

`copy` Makes an identical copy of an object of `Image` enforcing allocation of new memory for the image data. Note that in R, a simple assignment like `a<-b` does lead to copying of the data until either `a` or `b` are further modified.

`combine` Acts similarly to `rbind` and `cbind`. It allows to combine images to stacks adding further images at the back of the first one. Properties of the first image in the argument `x` are transferred to the result. All images must be of the same size (in first two dimensions) and color mode. If applied to a list of images, it calls `do.call("combine", x)`

`header` Acts similarly to copy, but does not copy the actual image data, only all the other slots. This function can be useful for creating new images from existing large ones preserving attributes.

`is.Image` Returns `TRUE` if argument is a valid `Image` and `FALSE` otherwise.

`assert` Compares dimensions and color modes of two images. If argument `strict` is `FALSE` images are allowed to have different number of frames.

Value

The constructors `Image`, `copy`, `combine` and `header` return a new object of `Image`.

`assert` and `is.Image` return a `logical`.

`stopIfNotImage` will return invisible `NULL` if its argument is of `Image` and an error message otherwise.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2005-2007

See Also

`Image`, `Image`, `Image`

Examples

```
i1 = Image()
sx = exp(24i*pi*seq(-1, 1, length=300)^2)
i2 = Image(outer(Im(sx), Re(sx)))
if (interactive()) display(normalize(i2))
i3 <- copy(i2)
is.Image(i2)

## see 'stackObjects' for example on combine
```

denoise *Noise removal, blurring and smoothing or images*

Description

This set of functions allows for the removal of noise, blurring and smoothing of images. The functions operate of images in any image mode. The functions and the corresponding help descriptions are ported from *ImageMagick*, see the reference below.

Usage

```
# Noise removal:
## S4 method for signature 'Image':
denoise(x, r=0, ...)
## S4 method for signature 'Image':
mediansmooth(x, r=2, ...)
## S4 method for signature 'Image':
despeckle(x, ...)

# Sharpening images:
## S4 method for signature 'Image':
sharpen(x, r=0, s=0.5, ...)
## S4 method for signature 'Image':
umask(x, r=0, s=0.5, amount=5, t=2, ...)

# Blurring images:
## S4 method for signature 'Image':
blur(x, r=0, s=0.5, ...)
## S4 method for signature 'Image':
gblur(x, r=0, s=0.5, ...)

# Adding noise to images:
## S4 method for signature 'Image':
noise(x, type="G", ...)
```

Arguments

x	An object of Image.
r	A numeric value for the radius of the pixel neighbourhood. Passing 0 enables automatic radius selection, default.
s	A numeric value for the standard deviation of the Laplacian (sharpen) or Gaussian (umask, blur, gblur), in pixels. For reasonable results, in most functions r must be larger than s.
amount	A numeric value for the percentage difference between the original and the blurred image that is added back into the original in the un-sharp mask algorithm.
t	A numeric value for the threshold in pixels needed to apply the amount in the un-sharp mask algorithm.

`type`	The type of noise to add. Supported noise types are: `Uniform`, `Gaussian` (default), `Multi`, `Impulse`, `Laplace` and `Poisson`. The value can be specified by one letter. Case insensitive.
`...`	Reserved.

Details

`despeckle` reduces the speckle-type, single-pixel, noise.

`mediansmooth` smooths the noisy image by replacing each pixel by a median of pixel values in taken over the neighbouring as defined by radius.

`blur`, `gblur` produce a blurred image. The `blur` method differs from the Gaussian blur, `gblur`, in that it uses a separable kernel which is faster but mathematically equivalent to the non-separable kernel.

`sharpen`, `umask` sharpen an image. `umask` uses the un-sharp mask algorithm, in which the image is convolved with a Gaussian operator of the given radius and standard deviation, `s`.

Value

A transformed image in an object of `Image`.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2005-2007

References

ImageMagick: `http://www.imagemagick.org`.

See Also

`Image`, `Image`

Examples

```
w <- 120
a <- Image((0:(w^2))/w^2, c(w,w))
if ( interactive() ) display(a)
b <- normalize(noise(a) * 0.1)
if ( interactive() ) display(b)
dn <- despeckle(b)
if ( interactive() ) display(dn)
bl <- blur(dn, 4, 2)
if ( interactive() ) display(bl)
```

display *Interactive image display*

Description

Display images on the screen of a local or remote display.

Usage

```
display(x, no.GTK=FALSE, ...)

## S4 method for signature 'Image':
animate(x, ...)        ## not available on Windows
## S4 method for signature 'IndexedImage':
animate(x, ...) ## not available on Windows
## S4 method for signature 'array':
animate(x, ...)        ## not available on Windows
```

Arguments

x	An object of Image, IndexedImage or array.
no.GTK	A logical value, if TRUE an *ImageMagick* display will be used instead of the GTK display (read details below). On Windows the *ImageMagick* display is not available.
...	Reserved.

Details

The argument main can be used with the GTK display to substitute the default window title. By default, the GTK display shows the expression used in call to display for the window title.

When used with IndexedImage display additionally accepts a colorize argument. If it is provided with any value of any class, then objects are mapped on the display using random RGB colors.

If available the GTK display will be used by default. This mode allows display of multiple images simultaneously. If the GTK mode is not used, an ImageMagick-internal display function, is used. This mode is for compatibility purposes only and should not be used if GTK is available. Because ImageMagick does not provide any programmatic ways to control and close display windows, the display in this mode is limited to one it time. The display window must be explicitly closed before a new image can be displayed. If EBImage was installed without GTK support, the function will fall back to the ImageMagick display automatically. In this case there is no need to supply no.GTK argument.

So far animate uses the AnimateImages function from ImageMagick, which behavior is analogous to that of the ImageMagick display. This function is not available on Windows due to the lack of support in the ImageMagick API.

GTK display does not use GTK widgets available from other R packages, the functionality is coded in EBImage. Therefore, the package must be compiled with GTK support in order to enable it.

Value

An invisible NULL.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2005-2007

References

ImageMagick: http://www.imagemagick.org; *GTK*: http://www.gtk.org; *GTK on Windows*: http://gladewin32.sf.net

See Also

Image, Image, Image, Image, Image

Examples

```
## load images of nuclei (seed points later)
f <- paste( system.file(package="EBImage"), "images/Gene1_G.tif", sep="/" )
ii = readImage(f)
## normalize images
ii = normalize(ii, separate=TRUE)
## segment
mask = thresh(ii, 25, 25, 0.02)
## refine segmentation with morphology filters
mk3 = morphKern(3)
mk5 = morphKern(5)
mask = dilate(erode(closing(mask, mk5), mk3), mk5)
## index objects with 'watershed'
io = watershed( distmap(mask), 1.5, 1)
if (interactive()) {
  display(io)
  display(io, main="Watershed segmentation", colorize=T)
}
```

---

distmap *Distance map transform of binary images*

Description

The function computes a distance map transformation of a binary image, i.e. of an image whose pixels are labeled as *foreground* and *background*. In the distance map, each pixel contains the distance from that pixel to the nearest background pixel, or to the border of the image. Note: this is a trivial brute force implementation. Please look elsewhere for efficient implementations; see also the references.

Usage

```
## S4 method for signature 'Image':
distmap(x, t=0.05, exact=FALSE, bg=0.05, ...)
```

Arguments

x	A Grayscale object of Image. x is considered as a binary image, consisting of 0's for background and all other values for foreground.
t	A numeric vector of length 1, the background threshold in the range [0,1). Pixels with intensity smaller than t are considered as background.
bg	A numeric numeric vector of length 1, the allowed minimum of the fraction of background pixels in the image. Deprecated.
exact	???
...	Further arguments.

Value

A Grayscale object of Image with pixels containing the floor values of their distances to the nearest background points.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2006

References

2D Euclidean Distance Transform Algorithms: A Comparative Survey. R. Fabbri, L. da F. Costa, J.C. Torelli and O.M. Bruno. ACM Computing Surveys, Vol. 40, No. 1, Article 2 (Feb. 2008).

The Image Processing Handbook. John Russ. 5th edition, 2006, CRC Press.

VIGRA by Ullrich Koethe: http://kogs-www.informatik.uni-hamburg.de/~koethe/vigra

See Also

Image, Image, Image, Image

Examples

```
## Not run: see ?watershed for an example
```

---

drawtext *Drawing primitives on images*

Description

The family of functions to draw primitives on images. At the moment, there is only one functions out of the planned family to draw text.

Usage

```
## S4 methods for signatures 'img=Image,xy=numeric,label=character'
## and 'img=Image,xy=matrix,label=character' and
## S4 method for signature 'Image, list, list':
drawtext(img, xy, labels, font, col, ...)

drawfont(family=switch(.Platform$OS.type, windows="Arial", "helvetica"),
         style="n", size=14, weight=200, antialias=TRUE)
```

Arguments

img	An object of Image. With indexed images, please ensure that the color range is in [0,1].
xy	(x,y) coordinates of labels. For single frames a matrix with the first column being x and second y or a corresponding numeric vector as it would be used to construct such a matrix. For multiple frames, a list of corresponding matrices/vectors.
labels	A character vector of labels to be output. For multiple frames a list of such characters.
font	An S3 object of class DrawFont as returned by drawfont. If omitted, the drawfont functions is called internally to obtain the default values.
col	A character vector of font colors. One per frame, recycled between frames if required.
...	Reserved.
family	A character value for the font family to use. On Linux/UNIX machines one can try to use helvetica, times, courier and symbol. On Windows machines, one can specify installed TrueType fonts, like Arial.
style	A character value for the font style to use. Can be specified by providing the first letter only. Supported are: normal (default), italic, oblique.
size	A numeric value for the font size.
weight	A numeric value for the font weight (bold font). Supported values between 100 and 900.
antialias	A logical value for whether the font should be anti-aliased.

Value

An object of Image even if the supplied img was of any derived class. If supplied was an IndexedImage, ensure that it was normalized to the range $[0, 1]$ before calling drawtext: the function is for annotation purposes only and using it with IndexedImage's is likely to destroy indexing information!

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2007

See Also

Image

Examples

```
## FIXME: This example is currently excluded from MacOS builds as it fails in
## ImageMagick string assertion. Although I tried to find the reason for
## the problem, I do not have a test environment (MacOS). Please test it and
## report what goes wrong here and at which stage.

if ( length(grep("apple", Sys.getenv("R_PLATFORM"))) == 0 ) {
  ## load images
  f <- paste( system.file(package="EBImage"), "images/Gene1_G.tif", sep="/" )
  ii = normalize(readImage(f), separate=TRUE)
```

```
  ## segment
  mask = thresh(ii, 25, 25, 0.02)
  mk3 = morphKern(3)
  mk5 = morphKern(5)
  mask = dilate(erode(closing(mask, mk5), mk3), mk5)
  ## index objects in images and remove bad ones
  io = watershed( distmap(mask), 1.5, 1)
  ft = hullFeatures(io)
  mf = moments(io, ii)   ## need these for intensity and size
  for ( i in seq_along(ft) ) ft[[i]] = cbind(ft[[i]], mf[[i]])
  io = rmObjects(io, lapply(ft, function(x)
         which(x["h.s"] < 150 | x["int"] < 50 | 0.3 * x["h.p"] < x["h.edge"] )
       ))
  ft = hullFeatures(io)
  ## get centres of objects (list, for ii is a stack of 4)
  xy <- lapply(ft, function(x) x[,1:2])
  # create labels for objects (list, for ii is a stack of 4)
  labels <- lapply(xy, function(x) as.character(1:nrow(x)))
  ## set font properties: semi bold
  f <- drawfont()
  f$weight=600
  ## draw annotations, recycle 2 colours between 4 images
  annot <- drawtext(channel(ii,"rgb"), xy, labels, font=f, col=c("#F0B769","#ACEE3F"))
  if (interactive()) display(annot)
}
```

`enhance` *Enhancing images and colors*

Description

Functions to enhance and modify colors in images.

Usage

```
## S4 method for signature 'Image':
cgamma(x, level=1, ...)
## S4 method for signature 'Image':
contrast(x, sharpen=TRUE, ...)
## S4 method for signature 'Image':
enhance(x, ...)
## S4 method for signature 'Image':
equalize(x, ...)
## S4 method for signature 'Image':
modulate(x, value=100, ...)
```

Arguments

`x`	An object of `Image`.
`level`	A numeric for the gamma level.

sharpen	A logical specifying whether the contrast should be increased (TRUE) or decreased (FALSE).
value	A percent change in brightness, saturation, and hue. The default value to keep the values unchanged.
...	Reserved.

Details

cgamma gamma-corrects image. The same image viewed on different devices will have perceptual differences in the way the image's intensities are represented on the screen. Adjust all three channels with the level parameter. Values typically range from 0.8 to 2.3.

contrast enhances the intensity differences between the lighter and darker elements of the image.

enhance applies a digital filter that improves the quality of a noisy image.

equalize applies a histogram equalization to the image.

modulate lets you control the brightness, saturation, and hue of an image. Modulate represents the brightness, saturation, and hue as one parameter.

Value

A transformed image in an object of Image.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2006-2007

References

ImageMagick: http://www.imagemagick.org.

See Also

Image

---

edgeFeatures *Extraction of edge profiles and edge features from images of indexed objects*

Description

Edge profile is a distance profile from the geometric center of the object to all its perimeter points calculated by taking the distance at different rotation angles (rotation around the center). The profile is calculated for the rotation angle, theta in [-pi,pi].

Usage

```
## S4 method for signature 'IndexedImage':
edgeProfile(x, ref, n=32, fft=TRUE, scale=TRUE, rotate=TRUE, ...)
## S4 method for signature 'IndexedImage':
edgeFeatures(x, ref, ...)
```

Arguments

x	An object of IndexedImage.
ref	A reference Grayscale image of the same size as x. See details.
n	An integer value giving the number of angle measures. The full circle of [-pi,pi] is divided into n-1 segments, at which edges the profile is approximated.
fft	A logical value. If TRUE, the resulting profile is the fft transformation of the distance profile giving the frequences of angular changes in shape.
scale	A logical value. If TRUE, the resulting profile is scaled by the effective radius (calcualted as part of link{hull.features}) making the profile scale invariant.
rotate	A logical value. If TRUE, the resulting profile is shifted by the object's roation angle (calculated from the moments on the ref image, if provided, and on the hull otherwise.
...	Reserved.

Details

The ref image can be omitted. However, if supplied it affects the centers of objects and their angles of rotation as in moments.

The edge.features runs with n=16 calculating scale and rotation invariant features. From the original distance profile, it returns the measure of object's irregularity taken by calculating the difference between the farthest and the closest to the center edge points. Then it computes the fft and returns 4 of it's lowest frequences, corresponding to 2*Pi, 2*Pi/2=Pi, 2*Pi/3 and 2*Pi/4=Pi/2.

The extracted feature names carry a e. prefix to indicate edge features.

Value

For a single frame, both functions return a matrix of descriptors with objects in rows and ordered profile points (or features) in columns. For image stacks, a list of such matrices.

The matrix columns in edge.profile correspond, from left to right, to the equidistant divisions of the range [-pi,pi] if fft is not used, otherwise to the frequences of angular changes (lower frequences on the left, higher on the right up to the middle of the vector).

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2007

See Also

IndexedImage, IndexedImage, IndexedImage, IndexedImage, IndexedImage

Examples

```
## see example(getFeatures)
```

haralickMatrix *Co-occurrence matrices (GLCM) and Haralick texture features*

Description

A set of functions to compute the co-occurrence matrix (GLCM - gray level co-occurrence matrix) and haralick texture features of objects in an indexed image.

Usage

```
## S4 method for signature 'IndexedImage, Image':
haralickMatrix(x, ref, nc = 32, ...)
## S4 method for signature 'IndexedImage, Image':
haralickFeatures(x, ref, nc = 32, ...)
```

Arguments

x	An object of IndexedImage used as a mask.
ref	A Grayscale object of Image-class. The texture is calculated from the data in this image.
nc	A numeric value. Specifies the number of gray levels to separate ref into when calculating the co-occurrence matrix. Defaults to 32
...	Reserved.

Details

haralickMatrix computes a co-occurrence matrix of dimension nc * nc for each object in an image. The co-occurrence matrix is constructed by assigning to element [i,j] the number of times a pixel of value i is adjacent to a pixel of value j.

In order to achieve rotational invariance the co-occurrence matrix is computed using four passes through each object. Thus each pixel is compared to four neighbours: right, below, diagonally down left and diagonally down right. In order to achieve symmetry in the co-occurrence matrix the reciprocal of each of these scores is also added. Thus for each pixel we have have a score for comparison with each of the surrounding pixels.

Finally the entire matrix is divided by the total number of comparisons, giving a probability for each of the possible combinations.

haralickFeatures computes for each object in an image 12 Haralick texture features, calculated from the co-occurrence matrix. The feature names carry a t. prefix (t for texture). The features and their calculations are:

asm Angular second moment: sum[_i=1^nc] sum[_j=1^nc]p(i,j)^2.

con Contrast: sum[_i=2^(2*nc)] n^2 * sum[_i=1^nc] sum[_j=1^nc] p(i,j), for all i,j s.t ABS(i - j) = n.

cor Correlation of GLCM: sum[_i=1^nc] sum[_j=1^nc]((i * j) * p(i,j) - mu_x * mu_y) / sigma_x * sigma_y.

var Variance: sum[_i=1^nc] sum[_j=1^nc](i - mu)^2. * p(i,j).

idm Inverse difference moment: `sum[_i=1^nc] sum[_j=1^nc] p(i,j) / (1 + (i - j)^2) .`

sav Sum average: `sum[_i=2^(2*nc)] i * Px+y(i).`

sva Sum variance: `sum[_i=2^(2*nc)] (i - sen)^2 * Px+y(i).`

sen Sum entropy: `-sum[_i=2^(2*nc)] Px+y(i) * log( p(i,j) ).`

ent Entropy: `-sum[_i=1^nc] sum[_j=1^nc] p(i,j) * log( p(i,j) ).`

dva Difference variance: `sum[_i=0^(nc-1)] (i^2) * Px-y(i).`

den Difference entropy: `sum[_i=0^(nc-1)] Px-y(i) * log( Px-y(i,j) ).`

f12 Measure of correlation: `ABS(ent - HXY1) / HX.`

f13 Measure of correlation: `sqrt( 1 - exp( -2 *(HXY2 - ent) ) ).`

Where:

p(i,j) The value in row i, column j of the co-occurrence matrix.

Px(i) Partial probability density function. Defined by `sum[_j=1^nc] p(i,j) .`

Py(j) Partial probability density function. Defined by `sum[_i=1^nc] p(i,j) .`

mu_x, mu_y Are the means of `Px` and `Px`, the partial probability density functions.

sigma_x, sigma_y Are the standard deviations of `Px` and `Py`.

Px+y Is the probability of the co-occurrence matrix co-ordinates summing to `x+y`. It is defined as `Px+y(k) = sum[_i=1^nc] sum[_j=1^nc] p(i,j), i + j = k and k = 2,3,...,2*nc.`

Px-y Is the probability of the absolute value of the difference between co-occurrence matrix co-ordinates being equal to `x-y`. It is defined as `Px-y(k) = sum[_i=1^nc] sum[_j=1^nc] p(i,j), ABS(i - j) = k and k = 2,3,...,2*nc.`

HXY1 `-sum[_i=1^nc] sum[_j=1^nc] p(i,j) * log( Px(i),Py(j) ) .`

HXY2 `-sum[_i=1^nc] sum[_j=1^nc] Px(i)*Py(j) * log( Px(i),Py(j) ) .`

Value

For a single frame in `x` the result of `haralickMatrix` is an array of dimensions `nc` * `nc` * "the number of objects in the frame".

`haralickFeatures` returns a matrix of dimensions "number of objects in the frame" * 12.

For multiple frames a list of each of the above will be returned.

Author(s)

Mike Smith, ⟨msmith@ebi.ac.uk⟩; Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2007

References

R. M. Haralick, K Shanmugam and Its'Hak Deinstein (1979). *Textural Features for Image Classification*. IEEE Transactions on Systems, Man and Cybernetics.

See Also

`IndexedImage`, `IndexedImage`, `IndexedImage`

Examples

```
## see example(getFeatures)
```

`hullFeatures` *Extraction of hull features from images of indexed objects*

Description

Hull features are a set of numeric descriptors of the hull of an object. These include coordinates, perimeter, size, acircularity etc.

Usage

```
## S4 method for signature 'IndexedImage':
hullFeatures(x, ...)
```

Arguments

`x`	An object of `IndexedImage`.
`...`	Reserved.

Details

The extracted features are (names carry an `h.` prefix to indicate hull features): `x,y` - coordinates of the geometric center, `s` - size (area), `p` - perimeter, `pdm` - mean distance to perimeter (from the center), `pdsd` - standard deviation of the distance to perimeter, `effr` - effective radius (is the radius of a circle with the same area), `acirc` - acircularity (fraction of pixels outside of the circle with `r=reff`), `sf` - shape factor (`per / ( 2 * sqrt(Pi * s))`), `edge` - number of pixels at the edge of the image, `theta` - hull's rotation angle (calculated without taking intensity values into account), `s2maj` - 2 times semi major (square root of the larger eigenvalue of the `moments` covariance matrix) - correlates with the distance from the center to the edge along the major axis, `s2min` - same but for the smaller eigenvalue (minor axis), `ecc` - eccentricity (`sqrt(eig1-eig2)/smaj`), `I1, I2` - first and second rotation invariant moments of the hull (as in `moments`).

Value

For a single frame, a matrix of descriptors with objects in rows and features in columns. For image stacks, a list of such matrices.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2007

See Also

`IndexedImage`, `IndexedImage`, `IndexedImage`, `IndexedImage`, `IndexedImage`

Examples

```
## see example(getFeatures)
```

moments *Image moments and moment invariants, feature extraction*

Description

A set of functions to compute central, scale and rotation invariant moments as well as to estimate object rotation angles and elongations based on central moments.

Usage

```
## S4 methods for signatures 'x=IndexedImage,ref=Image'
## and 'x=IndexedImage,ref=missing'
cmoments(x, ref, ...)

## S4 method for signature 'IndexedImage, Image':
smoments(x, ref, pw=3, what="s", ...)
## S4 method for signature 'IndexedImage, missing':
smoments(x, ref, pw=3, what="s", ...)

## S4 methods for signatures 'x=IndexedImage,ref=Image'
## and 'x=IndexedImage,ref=missing'
rmoments(x, ref, ...)

## S4 methods for signatures 'x=IndexedImage,ref=Image'
## and 'x=IndexedImage,ref=missing'
## and 'x=Image,ref=missing'
moments(x, ref, ...)
```

Arguments

x	An object of Image; a Grayscale object of Image in the last case.
ref	A Grayscale object of Image.
pw	A numeric value. When calculating central os scale invariant moments using smoments pw specifies the order of the matrix of moments to calculate.
what	A character string, or the first case insensitive letter, for the type of the moments to calculate: Central, Scale invariants, or Rotation invariants (specifying R is essentiall the same as calling rmoments.
...	Reserved.

Details

cmoments computes centers of the objects (x=M10/M00, y=M01/M00), their "mass" (intensity, int=M00) and the area (number of non-background pixels).

smoments computes for each object in an image a square matrix of order pw of either central moments (mu[i,j]) or scale invariant moments (nabla[i,j]) depending on the parameter what.

rmoments computes for each object the Hu's set of 7 rotation invariants.

`moments` computes for each object a summary of interesting moments and derived descriprots that include mass (total intensity), location, elements of the covariance matrix and its eigenvalues, rotation angle and the Hu's 7 rotation invariants.

Functions with signature `x=IndexedImage` work on images with multiple indexed objects in every frame. The index information is used to identify every object and serves as a mask resetting for every object pixels outside of the mask to background. The intensity is retrieved from the corresponding reference image. If it is not supplied, the intensity is set to 1 at each pixel of an object.

`moments` with `x='Image'` expect input image to contain one and only one object in each frame (such as those obtained by `stackObjects`).

Value

For `x=IndexedImage`:

For a single frame in `x` the result of `smoments` with types `S` and `C` is a 3D array with first two dimensions building square matrices of size `pw*pw` and the third dimension corresponds to the number of indexed objects in the frame. All other functions return 2D matrices with different moments or descriptors in columns and objects in rows.

For multiple frames a list of the above will be returned.

For `x=Image`:

The result is a 2D matrix of with different moments or descriptors in columns and objects in rows. (each frame is assumed to contain a single object).

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2007

References

M.K. Hu, *Visual Pattern Recognition by Moment Invariants*, IRE Trans. Info. Theory, vol. IT-8, pp.179-187, 1962

Image moments: http://en.wikipedia.org/wiki/Image_moments

See Also

`IndexedImage`, `IndexedImage`, `IndexedImage`

Examples

```
## see example(getFeatures)
```

`zernikeMoments` *Extraction of Zernike moments from images of indexed objects*

Description

Extraction of Zernike moments from images of indexed objects

Usage

```
## S4 method for signature 'IndexedImage, Image':
zernikeMoments(x, ref, N = 12, R = 30, apply.Gaussian=TRUE, pseudo=FALSE, ...)
```

Arguments

x	An object of IndexedImage.
ref	An object of Image-class in the Grayscale mode.
N	Integer value defining the degree of the Zernike polynomials, which in turn defines the number of features calculated. Defaults to 12.
R	Defines the radius of the circle around an object centre from which the features are calculated. It also defines the standard deviation for the 2D Gausian applied at the centre of an object. See details. Defaults to 30.
apply.Gaussian	A logical value that specifies if a local 2D Gaussian modification should be applied to every object at its centre. Defaults to TRUE.
pseudo	Specifies if an alternative algorithm should be used to calculate pseudo features. Defaults to FALSE.
...	Reserved.

Details

Zernike features are calculated as follows:

`Z_nl = (n+1) / pi * ABS( sum_x,y(V*nl(x,y) * i(x,y)) ),`

`0 <= l <= n, n - l` is even and `i(x,y)` is the intensity of the reference image at the cordinates `(x,y)` that fall withing a circle of radius `R` from the object's centre. Coordinates taken relative to the object's centre. The `ABS` of the sum gives a real value of the complex feature and makes it rotation invariant.

`V*nl` is a complex conjugate of a Zernike polynomial of degree `n` and angular dependence `l`:

`Vnl(x,y) = Qnl(x,y) * exp(j*l*theta)`, where `j = sqrt(-1), theta=atan2(y,x)`, and

`Qnl(x,y) = sum[_m=0^((n-l)/2)] ((-1)^m * (n-m)! * r^(n-2*m)) / (m! * ((n-2*m+l)/2)! * ((n-2*m-l)/2)!)` , where `r = sqrt(x^2+y^2)`.

The extracted features all carry a `"z"`-prefix, for Zernike, and have the indexing of the form `z.0402` where 04 in this is for `n=4` and 02 for `l=2`. The number of differnt `l`-values is calculated automatically from `N`. For a given `N` all combinations of `(n,l)` are calculated. For the default `N = 12` the resulting number of features is 49. Columns in the results matrix are firstly ordered by increasing value of `n` and secondly by increasing value of `l`.

If `apply.Gaussian = TRUE` then prior to calculating the Zernike features, a Gaussian is applied to the image centred at the co-ordinates of an objects centre of mass, with the standard deviation defined by `sigma = 0.8 * R`. The centres of mass are found by using `moments(x, ref)`. The resulting image is then:

`i'(x,y) = i(x,y) * exp( -(x^2+y^2) / (2 * sigma^2) ).`

The expected effect: the default value of `R = 30` and `sigma = 0.8 * R = 24` will penalize the edges of objects of radii more than 20 and intensity is expected to be around 0 at distances from the centre more than 40.

Value

For a single frame, a matrix of descriptors with objects in rows and features in columns. For image stacks, a list of such matrices. Frames with no objects will result in 0-populated a matrix for 1 object.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩; Mike Smith, ⟨msmith@ebi.ac.uk⟩, 2007

References

F. Zernike. *Beugungstheorie des Schneidenverfahrens und seiner verbesserten Form, der Phasenkontrastmethode (Diffraction theory of the cut procedure and its improved form, the phase contrast method)*. Physica, 1:pp. 689-704, 1934.

Jamie Shutler, *Complex Zernike Moments*: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/SHUTLER3/node11.html

See Also

IndexedImage, IndexedImage, IndexedImage, IndexedImage

Examples

```
## see example(getFeatures)
```

---

filter2 *2D Convolution Filter*

Description

2D convolution-based linear filter for images and matrix data.

Usage

```
## S4 method for signature 'Image, matrix':
filter2(x, filter, ...)

mkball(n, shape="step")
mkbox(n)
```

Arguments

x	An object of Image in Grayscale mode, a numeric array or a matrix.
filter	A square matrix with odd dimensions.
n	A positive integer of length 1, specifying the size (number of rows and columns) of the returned square matrix.
shape	A character vector of length 1, with one of two values: step for a step function, ball for a semisphere.
...	Further arguments.

Details

The convolution filter is based on `fft` transforms. If `x` is an `array`, the filter is applied per frame (as to images).

Value

`filter2` returns a transformed object of the same class as `x`.

`mkball` and `mkbox` return a square matrix that can be used for the `filter` argument of `filter2`. Its dimensions (number of rows and columns) are given by `n`. For `shape="step"`, the entries of the matrix are the integers 0 and 1, and the 1s correspond to the interior of a circle of radius `n/2` around the center element of the matrix. For `shape="ball"`, the non-zero elements of the matrix are `z = sqrt(radius^2 - x^2 - y^2)`, a parameterization of a semisphere.

Author(s)

Gregoire Pau, ⟨gpau@ebi.ac.uk⟩

See Also

`Image`, `Image`

`floodFill` *Filling matrix data with flood fill algorithm*

Description

Functions to fill regions in images and matrices/arrays and fill holes in indexed objects.

Usage

```
## S4 method for signature 'array':
floodFill(x, pt, col, tolerance=1e-3, ...)
## S4 method for signature 'IndexedImage':
fillHull(x, ...)
```

Arguments

`x`	An `array`-like object (e.g. `Image`) to be filled. For `fillHull` this must be an instance of `IndexedImage`.
`pt`	An integer of 2 values for `x-y` coordinates of the point where fill starts.
`col`	A value for the fill color, supposedly in the same storage mode as `x`. If `x` is an image `col` will be converted to the corresponding storage mode using the `channel` function, and thus can be a character value specifying the color in X11 mode.
`tolerance`	Color tolerance used during the fill.
`...`	Reserved.

Details

If no color is specified, the color under the coordinates of `pt` is used (this only has sense if tolerance is not zero). The flood fill is implemented using the fast scan line algorithm. It is assumed that `floodFill` is run either on matrices of single-framed images. If the latter is not the case, only the first frame will be modified!

`fillHull` files holes in objects of `IndexedImage`'s.

Value

A copy of `x` with the fill applied.

Author(s)

Gregoire Pau, Oleg Sklyar; 2007

See Also

`Image`, `Image`

Examples

```
if (interactive()) {
  data(imageWithHoles)

  display(x, main="Original image")
  display(floodFill(x,c(5,5),tolerance=0.2))
  display(floodFill(x,c(40,40),tolerance=0.2))

  mask = x
  mask[mask>0.3]=1
  mask[mask<=0.3]=0

  y = floodFill(mask,c(40,40),2)
  class(y) = "IndexedImage"
  display(y, main="Indexed mask with holes")

  y = fillHull(y)
  display(y, main="Indexed mask without holes")

  rgb = channel(normalize(y), "rgb")
  rgb = floodFill(rgb, c(1,1), "red")
  rgb = floodFill(rgb, c(50,50), "blue")
  rgb = floodFill(rgb, c(150,150), "green")
  display(rgb)
}
```

`frameDist` *Calculate pairwise distances between image frames or frames of two images*

Description

Given two images with multiple frames, `frameDist` calculates a matrix of distances between frames (the more similar are the images, the smaller are the distances). Prerequisite: images should be centered and rotationally and scale aligned. The function is recommended with `stackObjects` with arguments `rotate=combine=TRUE`.

Usage

```
## S4 method for signature 'Image, Image':
frameDist(x, y, r, g, b, blur=TRUE,
    method="dist", verbose, ...)
## S4 method for signature 'Image, missing':
frameDist(x, y, r, g, b, blur=TRUE,
    method="dist", verbose, ...)
```

Arguments

`x,y`	A image stacks.
`r,b,g`	If `x` and `y` are in TrueColor mode, these values specify weights of red, green and blue channels in the resulting distance.
`blur`	A logical indicating whether frames ought to be blurred before comparison (`TRUE` recommended).
`method`	Method to use: `dist` - Euclidian distance, `dot` - dot product.
`verbose`	Provides additional output as the function can be lengthy.
`...`	Reserved.

Details

For Grayscale images, the distance of each pair of frames, `xf` and `yf`, is calculated as `mean(abs(xf-yf))`, where `mean` is taken over the pixels with at least `xf` or `yf` being non-zero. For TrueColor images, the distance is calculated as `mean(sqrt(r*rc^2+g*gc^2+b*bc^2))`, where `rc,gc,bc` are `[0,1]` ranged values of red, green and blue channels and `r,g,b` are the weights. Again, the mean is taken only over non-zero pixels.

Value

A matrix of distances. If `y` is missing - a square symmetric matrix for the distances between frames in the same image with diagonal elements set at `Inf`.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2007

See Also

`stackObjects`

`getFeatures` *Feature extraction for objects in indexed images*

Description

Functions to extract numerical descriptors for objects in indexed images. The functions here call a set of individual feature extraction routines for every feature set like hull, texture (Haralick), edge, moment features etc.

Usage

```
## S4 method for signature 'IndexedImage':
getFeatures(x, ref, N=12, R=30, apply.Gaussian=TRUE, nc=256, ...)
## S4 method for signature 'IndexedImage':
features(x, ...)
```

Arguments

`x`	An object of `IndexedImage` (like those obtained from `propagate` or `watershed`.
`ref`	A reference image containing images of the objects in Grayscale mode (intensity values)!
`N`	Passed to `zernikeMoments`. Integer value defining the degree of the Zernike polynomials, which in turn defines the number of features calculated. Defaults to 12.
`R`	Passed to `zernikeMoments`. Defines the radius of the circle around an object centre from which the features are calculated. It also defines the standard deviation for the 2D Gausian applied at the centre of an object. See details. Defaults to 30.
`apply.Gaussian`	Passed to `zernikeMoments`. A logical value that specifies if a local 2D Gaussian modification should be applied to every object at its centre. Defaults to `TRUE`.
`nc`	Passed to `haralickFeatures`. A numeric value. Specifies the number of gray levels to separate `ref` into when calculating the co-occurrence matrix. Here defaults to 256.
`...`	Reserved.

Details

`ref` can be missing in calls to `getFeatures`. In this case texture and other intensity-dependent features will be omitted.

Value

`getFeatures` calculates feature matrices and returns an object of `IndexedImage` with its `features` slot set. The typical use would be to re-assign the result to the original image itself, i.e. `x <- getFeatures(x, a)`.

`features` extracts the content of the `features` slot of the `IndexedImage`. If it is empty, the function calls `getFeatures` without a reference image and returns the generated feature set. In this case the `features` slot of the original image is not updated.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2007

See Also

`IndexedImage`, `IndexedImage`, `IndexedImage`, `IndexedImage`, `IndexedImage`, `IndexedImage`, `IndexedImage`, `IndexedImage`, `IndexedImage`

Examples

```
## load images
f <- paste( system.file(package="EBImage"), "images/Gene1_G.tif", sep="/" )
ii = readImage(f)
## normalize images
ii = normalize(ii, separate=TRUE)
## segment
mask = thresh(ii, 25, 25, 0.02)
## refine segmentation with morphology filters
mk3 = morphKern(3)
mk5 = morphKern(5)
mask = dilate(erode(closing(mask, mk5), mk3), mk5)
## index objects with 'watershed'
io = watershed( distmap(mask), 1.5, 1)
ft = hullFeatures(io)
mf = moments(io, ii)   ## need these for intensity and size
for ( i in seq_along(ft) ) ft[[i]] = cbind(ft[[i]], mf[[i]])
io = rmObjects(io, lapply(ft, function(x)
       which(x[,"h.s"] < 150 | x[,"int"] < 50 | 0.3 * x[,"h.p"] < x[,"h.edge"] )
     ))
io = getFeatures(io, ii)
str( features(io) )
```

---

readImage *Image I/O*

Description

Functions to choose, read and write images from/to files and URL's. Supported image formats are determined by the ImageMagick installation.

Usage

```
chooseImage(colormode = Grayscale)
readImage(files, colormode = Grayscale, ...)

## S4 method for signature 'Image':
writeImage(x, files, ...)
```

Arguments

x	An object of Image.
files	A character vector of files/URL's to read from or to write to.
colormode	An integer value for the color mode of images after they are read. By default all images will be converted to Grayscale on read.
...	Reserved.

Details

If files is missing when writing images, fileName(x) method will be used to select a single file to write to. Otherwise, the length of this vector must be equal either 1 or the number of 2D images in the stack.

When writing images in formats supporting non-lossless compression (like JPEG), the quality can be spcified used a numeric quality argument with the range [1,100]. Higher values correspond to better quality. Defaults to 95.

The file format is deduced from the file name extension(s), there is neither a need nor a way to specify the format explicitly.

ImageMagick is used to perform all image I/O operations. Therefore, the package supports all the file types supported by ImageMagick.

When reading images, files of different formats can be mixed in any consequence, including mixing single 2D images with TIFF image stacks. The result will contain a stack of all images and stacks cropped (filled with background if images are smaller) at the size of the first image read.

choose.image is an interactive function that does not return to R until either Ok or Cancel button is pressed in the GUI dialog. It uses GTK+2 File Open Dialog to select images. Multiple images can be selected and loaded at once. By default this functions reads images as TrueColor. This function will produce an error message if the package was compiled without GTK+ support.

Value

For readImage and chooseImage a new instance of Image.

For writeImage an invisible NULL.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2005-2006

References

ImageMagick: http://www.imagemagick.org.

See Also

Image, Image, Image

Examples

```
  f <- paste( system.file(package="EBImage"), "images/Gene1_G.tif", sep="/" )
  ii = readImage(f)
if ( interactive() ) {
  url <- c("http://www.google.com/intl/en/images/logo.gif")
  im <- readImage (url, TrueColor)
```

```
  ## Not run: writeImage (im, "googlelogo.tif")
  ## Not run: im1 <- channel ( chooseImage(), "gray")
}
```

`matchObjects` *Matching objects in two indexed images*

Description

For objects detected in one image, this function finds indexes of matching obnects in the other (i.e. indexes of objects at the locations of the centres of objects in the original image.

Usage

```
## S4 method for signature 'IndexedImage, IndexedImage':
matchObjects(x, ref, ...)
```

Arguments

`x`	An object of `IndexedImage`.
`ref`	A reference image `IndexedImage` with objects to matched to.
`...`	Reserved.

Value

If number of frames in `x` and `ref` is 1, then the result is an `integer` vector of matching indexes, similar to `match`.

Otherwise, a list of such vectors, 1 per frame.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2007

See Also

`IndexedImage`, `IndexedImage`, `IndexedImage`

`closing` *Morphological operations on images*

Description

Functions to perform morphological operations on binary images.

Usage

```
## S4 method for signature 'Image':
dilate(x, kern=morphKern(5), iter=1, ...)
## S4 method for signature 'Image':
erode(x, kern=morphKern(5), iter=1, ...)
## S4 method for signature 'Image':
opening(x, kern=morphKern(5), iter=1, ...)
## S4 method for signature 'Image':
closing(x, kern=morphKern(5), iter=1, ...)

morphKern(size=5, shape="round")
```

Arguments

`x`	An object of `Image`. `x` should be a binary image in the `Grayscale` mode. If image is not binary, all non-zero pixels will be considered as 1 to turn the image into a binary `{0,1}` image.
`kern`	Kernel mask matrix.
`iter`	Number of iterations.
`size, shape`	Kernel matrix size and shape.
`...`	Reserved.

Details

`morphKern` can be used to generate a kernel matrix for the use with any of the morphological operators. The function can generate round and square kernels of odd size, e.g. 5, 7, 9 etc. Even sizes are not supported because the location of the centre pixel is undefined. Kernels can be altered in any desired way, this is just a convenience function.

`erode` applies the mask positioning its centre over every background pixel (0), every pixel which is not covered by the mask is reset to foreground (1). In this way image features grow in size.

`dilate` applies the mask positioning its centre over every foreground pixel (1), every pixel which is not covered by the mask is reset to background (0). In this way image features seem shrink in size.

`opening` is erosion followed by dilation and `closing` is dilation followed by erosion.

Value

A transformed image in an object of `Image`.

`morphKern` returns a square matrix of 0 and 1 of a given size.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2006

References

ImageMagick: http://www.imagemagick.org.

See Also

Image, Image, Image

Examples

```
## see example(propagate)
```

---

normalize *Functions to normalize images*

Description

Functions to normalize images.

Usage

```
## S4 method for signature 'Image':
normalize(x, separate=TRUE, ft=c(0,1), ...)

## S4 methods for signature 'x=Image'
negate(x, ...)
normalize2(x, ...)
```

Arguments

x	An object of Image-class.
separate	If TRUE normalizes each frame separately.
ft	A numeric vector of 2 values, target minimum and maximum intensity values after normalization for Grayscale images.
...	Reserved.

Details

negate negates the colors in the reference image. Operates on all image modes.

normalize normalizes Grayscale images to the given range.

normalize2 uses ImageMagick normalization routine to normalize Grayscale or TrueColor images.

Value

A transformed image in an object of Image.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2006-2007

References

ImageMagick: http://www.imagemagick.org.

See Also

Image, Image

Examples

```
## see example(getFeatures)
```

---

paintObjects *Marking detected objects in reference images*

Description

This function allows to mark objects detected with getFeatures or watershed in colour for preview.

Usage

```
## S4 method for signature 'IndexedImage, Image':
paintObjects(x, tgt, opac=c(0.4, 0.05, 0.4), col=c("#FFC72C","#5BABF6","#FF372
```

Arguments

x	An object of IndexedImage in the Grayscale mode with integer-abased object encoding, as returned by watershed.
tgt	A reference grayscale image to calculate object intensity. Should be TrueColor to produce coloured output.
opac	A numeric vector of opacity values for foreground (object boundary), object background and edges of object contacts. At least 3 values in the above sequence must be supplied. Opacity range is [0,1] with 0 being fully transparent.
col	A character vector of full colours (before opacity applied), colour names supported, to draw object boundaries, object background and edges of object contacts. At least 3 values must be supplied. Default color scheme is yellow for edges, blue for background and red for object contacts and object on image edges.
...	Reserved.

Value

A copy of tgt in the same colour mode with objects marked on top of the image. features of x are not transfered – this result is for visualization only.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2006-2007

See Also

IndexedImage, IndexedImage, IndexedImage, IndexedImage

Examples

```
## load images of nuclei (seed points later)
f <- paste( system.file(package="EBImage"), "images/Gene1_G.tif", sep="/" )
ii = readImage(f)
## normalize images
ii = normalize(ii, separate=TRUE)
## segment
mask = thresh(ii, 25, 25, 0.02)
## refine segmentation with morphology filters
mk3 = morphKern(3)
mk5 = morphKern(5)
mask = dilate(erode(closing(mask, mk5), mk3), mk5)
## index objects with 'watershed'
io = watershed( distmap(mask), 1.5, 1)

## load images of cells (the ones to segment with propagate)
f <- paste( system.file(package="EBImage"), "images/Gene1_R.tif", sep="/" )
xi = readImage(f)
## normalize images
xi = normalize(xi, separate=TRUE)
## segment
mask = thresh(xi, 40, 40, 0.0)
## refine segmentation with morphology filters
mk7 = morphKern(7)
mask = dilate(erode(closing(mask, mk7), mk5), mk7)
## index objects of xi with 'propagate' using ii as seeds
xo = propagate(xi, io, mask, 1e-5, 1.5)

## create an RGB preview of a combination of ii and xi
rgb = channel(ii, "asred") + channel(xi, "asgreen")
## paint cells on the preview
rgb = paintObjects(xo, rgb)
## paint nuclei on the preview
rgb = paintObjects(io, rgb)
if (interactive()) display(rgb)
```

---

propagate *Voronoi-based segmentation on image manifolds*

Description

R implementation of the Voronoi-based image segmentation on image manifolds [2].

Usage

```
## S4 method for signature 'Image, IndexedImage':
propagate(x, seeds, mask=NULL, lambda=0.1,
   ext=1, seed.centers=FALSE, ...)
```

Arguments

x	An object of Image to be segmented, in the Grayscale mode.
seeds	An object of IndexedImage of the same size as x in all three dimensions. This image provides seed points for object detection.
mask	An object of Image of the same size as x in all three dimensions; in the Grayscale mode. All zero regions will be excluded from object detection.
lambda	A numeric value. The regularisation parameter for the distance calculations, determines the trade-off between the Euclidian distance in the image plane and the contribution of the gradient of the values in x. See details.
ext	Extension of the neighborhood to estimate image gradient, in pixels in every direction from the central point, i.e. ext=1 means a 3x3 neighborhood.
seed.centers	If TRUE, only centers of the seed points are left in the seeds image supplied to the propagate algorithm.
...	Reserved.

Details

The method operates by computing a discretized approximation of the Voronoi regions for given seed points on a manifold with a metric controlled by local image features.

The metric is a Riemannian metric defined in terms of the image I and a regularization parameter lambda. With this metric the distance between pixels used to let the given seeds grow outwards (propagate) is

```
d^2 = (grad(I)^2 + lambda * (dx^2 + dy^2)) / (lambda + 1)
```

The above formulation was proposed by Carpenter et al, however in the calculation we use a modified distance measture, in which sharp gradients are downregulated and large distances additionally penalized. Effectively, we use the following formula:

```
d = sqrt(grad(I)) + 1e-3*lambda * (dx^2+dy^2)^2
```

The denominator is left out for speed reasons, so is the square root of the distance.

The gradient is calculated on a neighborhood of pixels (the width of which is controlled by the argument ext) to avoid relying on single (noisy) pixels. Lambda controls the weight of the Euclidian distance term. In case of large lambda, d turns into Euclidian distance in the (x,y)-plane. For small lambda, the distance will be dominated by the intensity gradient.

Value

An image of Image, with the same object indexing as seeds. No new objects are created, only those specified by seeds are propagated. Use getFeatures to assign the feature matrix.

License

The underlying C++ code is based on code from CellProfiler [1,3]. An LGPL license was granted by Thouis Jones to use this part of CellProfiler's code for the `propagate` function.

Author(s)

Original CellProfiler code: Anne Carpenter <carpenter@wi.mit.edu>, Thouis Jones <thouis@csail.mit.edu>, In Han Kang <inthek@mit.edu>.

Port for this package: Oleg Sklyar <osklyar@ebi.ac.uk> and Wolfgang Huber <huber@ebi.ac.uk>.

References

[1] *A. Carpenter, T.R. Jones, M.R. Lamprecht, C. Clarke, I.H. Kang, O. Friman, D. Guertin, J.H. Chang, R.A. Lindquist, J. Moffat, P. Golland and D.M. Sabatini*, "CellProfiler: image analysis software for identifying and quantifying cell phenotypes", Genome Biology 2006, 7:R100

[2] *T. Jones, A. Carpenter and P. Golland*, "Voronoi-Based Segmentation of Cells on Image Manifolds", CVBIA05 (535-543), 2005

[3] CellProfiler: http://www.cellprofiler.org

See Also

`IndexedImage`, `IndexedImage`, `IndexedImage`, `IndexedImage`

Examples

```
## load images of nuclei (seed points later)
f <- paste( system.file(package="EBImage"), "images/Gene1_G.tif", sep="/" )
ii = readImage(f)
## normalize images
ii = normalize(ii, separate=TRUE)
## segment
mask = thresh(ii, 25, 25, 0.02)
## refine segmentation with morphology filters
mk3 = morphKern(3)
mk5 = morphKern(5)
mask = dilate(erode(closing(mask, mk5), mk3), mk5)
## index objects with 'watershed'
io = watershed( distmap(mask), 1.5, 1)
if (interactive()) display(io)

## load images of cells (the ones to segment with propagate)
f <- paste( system.file(package="EBImage"), "images/Gene1_R.tif", sep="/" )
xi = readImage(f)
## normalize images
xi = normalize(xi, separate=TRUE)
## segment
mask = thresh(xi, 40, 40, 0.0)
## refine segmentation with morphology filters
mk7 = morphKern(7)
mask = dilate(erode(closing(mask, mk7), mk5), mk7)
## index objects of xi with 'propagate' using ii as seeds
xo = propagate(xi, io, mask, 1e-2, 2)
if (interactive()) display(xo)
```

`rmObjects` *Object removal*

Description

The `rmObjects` functions deletes objects indexed by a list of integer vectors (indexes) from an image with indexed objects. `reenumerate` re-enumerates all objects in an IndexedImage from 0 - background to the actual number of objects.

Usage

```
## S4 methods for signatures 'x=IndexedImage,index=numeric'
## and 'x=IndexedImage,index=list'
rmObjects(x, index, ...)

## S4 method for signature 'IndexedImage':
reenumerate(x, ...)
```

Arguments

`x`	An object of `IndexedImage`.
`index`	A numeric (integer) vector of indexes of objects to remove in the frame if `x` contains one frame only. For multiple frames, a list of such vectors.
`...`	Reserved.

Value

An image of `IndexedImage`.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2006-2007

See Also

`IndexedImage`, `IndexedImage`, `IndexedImage`, `IndexedImage`

Examples

```
## see example(drawtext)
```

segment *Segmentation and edge detection*

Description

Fucntions to segment images and detect edges.

Usage

```
## S4 method for signature 'Image':
edge(x, r=0, ...)
## S4 method for signature 'Image':
segment(x, cl=10, s=1.5, ...)
```

Arguments

x	An object of Image.
r	The radius of the pixel neighbourhood to take into account. The 0 value enables automatic radius selection.
cl	Minimum cluster size in pixels .
s	The smoothing threshold.
...	Reserved.

Details

edge returns an image of edges between different colours in the original image. Most effective on binary images.

segment segment an image by analyzing the histograms of the color components and identifying units that are homogeneous with the fuzzy C-means technique (source and implementation ImageMagick). The smoothing threshold eliminates noise in the second derivative of the histogram. As the value is increased, you can expect a smoother second derivative.

Value

A transformed image in an object of Image.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2005-2006

References

ImageMagick: http://www.imagemagick.org.

See Also

Image

stackObjects *Generate a stack of images for detected objects, one object per image*

Description

From an indexed image the functions generates an image stack with one image per object placing each object in the middle of the image. Objects can be automatically rotated to align them along the horizontal axis.

Usage

```
## S4 methods for signatures 'x=IndexedImage,ref=Image,index=character'
## and 'x=IndexedImage,ref=Image,index=list' and
## S4 method for signature 'IndexedImage, Image,
##   numeric':
stackObjects(x, ref, index, ...)

## S4 method for signature 'IndexedImage, Image,
##   missing':
stackObjects(x, ref, index, combine, rotate, bg.col, ext, centerby, rotateby,
```

Arguments

x	An object of IndexedImage. Images must be Grayscale and carry object indexing information, like those returned by watershed or propagate.
ref	A reference image containing images of the objects to be stacked using x both as index and the mask. Can be in any color mode.
index	Indexes of objects to stack. See details for supported types.
combine	Called on a stack if images, specifies if the resulting list of image stacks with individual objects should be combined into a single image stack. Defaults to FALSE.
rotate	Specifies if the objects should be aligned rotationally. Defaults to TRUE.
bg.col	Color for pixels outside the mask defined by x. Defaults to TRUE.
ext	Extension of the target bounding box. See details. If not given, ext is calculated from data.
centerby,rotateby	If ref is a TrueColor image, these character values specify which channel should be used to center and rotate objects. By default the image is converted by channel(ref,"gray"). Other possible values are "red", "green" or "blue".
...	Reserved.

Details

The bounding box is set to be a square, which centre coinsides with the geometric center of the object. The ext argument can be used to specified its size, where edge length will be given by 2*ext+1. If ext is not specified, it is calculated from the data by taking the 95% quantile of the vector of h.s2major descriptor of hullFeatures taken over all, objects in the image. This descriptor specifies the extension of the object along its major axis starting from its center.

The size of the bounding box is fixed for all frames to enable combining the resulting frames into a single multiframe image.

`index` must be coersable to numeric to specify object indexes that are taken into the resulting stack. It can be specified as `numeric` directly only for images with 1 frame. For images with multiple frames it can be either a `list` or a `character`. If specified as a `list`, it can be a named list of numeric indexes where names are converted to `character` frame indexes in an arbitrary order, or it can be an unnamed list of the same length as the number of frames. If specifies as `character`, each element must contain two numbers separated by a dot where the first number is the index of the frame and the second one is the index of the object within the frame, e.g. `"2.035"` will specify frame 2 and object 35.

Value

An image stack or a list of image stacks if `x` was itself a stack (contained more than one image).

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2006-2007

See Also

`tile`, `tile`, `tile`

Examples

```
## load images
f <- paste( system.file(package="EBImage"), "images/Gene1_G.tif", sep="/" )
ii = readImage(f)
## normalize images
ii = normalize(ii, separate=TRUE)
## segment
mask = thresh(ii, 25, 25, 0.02)
## refine segmentation with morphology filters
mk3 = morphKern(3)
mk5 = morphKern(5)
mask = dilate(erode(closing(mask, mk5), mk3), mk5)
## index objects with 'watershed'
io = watershed( distmap(mask), 1.5, 1)
if (interactive()) display(io)

## stack individual objects
s = stackObjects(io, ii, combine=FALSE)
## display stack of objects of from the first image
if (interactive()) display(s[[1]])
## combine stacks of objects into a single stack
s = combine(s)
## tile the stack into one frame
t = tile(s)
if (interactive()) display(t)
```

thresh *Image and color channel thresholding*

Description

Functions to threshold images and color channels of images.

Usage

```
## S4 method for signature 'Image':
thresh(x, w=5, h=5, offset=0.01, ...)
## S4 method for signature 'Image':
athresh(x, w=10, h=10, offset=0, ...)
## S4 method for signature 'Image':
cthresh(x, threshold=0, ...)
```

Arguments

x	An object of Image. For thresh it must be in the Grayscale mode.
w, h	Thresholding frame width and height in pixel.
offset	Threshold offset from the mean value.
threshold	Threshold value for channel thresholding (uniform).
...	Reserved.

Details

If thresh and athresh are adaptive thresholind functions. While athresh can be used on both Grayscale and TrueColor images, thresh can only be used on Grayscale images, however it is significantly faster on them.

The value of offset in thresh is on the same scale as data, i.e. the default value is selected assuming data in the range [0,1]. This value for athresh should be selected empirically, as this function uses AdaptiveThreshold of ImageMagick and the scale of this parameter was not documented; the reasonable values are usually in the orders of 500, 1000 or avobe (at least to obtain similar thresholding to that of 0.01 with thresh on cytomicroscopic images.).

Value

A new instance of Image in the same color mode as input.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2005-2007

References

ImageMagick: http://www.imagemagick.org.

See Also

Image, Image, Image, Image, Image

Examples

```
## see example(watershed)
```

---

tile *Generate a tiled image from a stack of images*

Description

Given an image stack (or a list of such those), tile generates for each stack a single image with frames tiled. untile does exactly the opposite dividing a tiled image into a stack.

Usage

```
## S4 method for signature 'Image':
tile(x, nx=10, lwd=1, fg.col="#E4AF2B", bg.col="gray", ...)
## S4 method for signature 'list':
tile(x, nx=10, lwd=1, fg.col="#E4AF2B", bg.col="gray", ...)
## S4 method for signature 'Image, numeric':
untile(x, nim, lwd=1, ...)
```

Arguments

x	An object of Image or a list of such objects. Images must be grayscale and carry object indexing information, like those returned by watershed or propagate.
nx	The number of tiled images in a row.
lwd	The width of the grid lines between tiled images, can be 0.
fg.col	The color of the grid lines (if lwd > 0). This will be converted to Grayscale if the color mode of x is Grayscale.
bg.col	The color of the background for extra tiles. This will be converted to Grayscale if the color mode of x is Grayscale.
nim	A numeric vector of 2 elements for the number of images in both directions.
...	Reserved.

Details

tile for x=list is a useful addition to stackObjects, which returns a list of stacks, thus it can be directly used on the result of the latter.

Value

An image of the same class and in the same color mode as x.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2006-2007

See Also

stackObjects, stackObjects

Examples

```
## see example(stackObjects)
```

`resize` *Image transformation: rotation, resize, etc.*

Description

Functions to rotate, mirror and resize images.

Usage

```
## S4 method for signature 'Image':
affinet(x, sx=0, rx=0, ry=0, sy=0, tx=0, ty=0, ...)
## S4 method for signature 'Image':
flip(x, ...)
## S4 method for signature 'Image':
flop(x, ...)
## S4 method for signature 'Image':
resample(x, w, h, ...)
## S4 method for signature 'Image':
resize(x, w, h, blur=1, filter="Lanczos", ...)
## S4 method for signature 'Image':
rotate(x, angle=90, col, ...)
```

Arguments

`x`	An object of `Image`.
`sx, rx, ry, sy, tx, ty`	Elements of the affine matrix.
`w, h`	Width and height of a new resized/resampled image. One of these arguments can be missing to enable proportional resize.
`blur`	The blur factor, where 1 (`TRUE`) is blurry, 0 (`FALSE`) is sharp.
`filter`	Resize pixel sampling filter.
`angle`	Image rotation angle in degrees.
`col`	A numeric, integer or character specifying the background color of the rotated image. Not implemented yet, defaults to black.
`...`	Reserved.

Details

`affinet` transforms an image as dictated by the affine matrix.

`flip` creates a vertical mirror image by reflecting the pixels around the central x-axis.

`flop` creates a horizontal mirror image by reflecting the pixels around the central y-axis.

`resample` scales an image to the desired dimensions with pixel sampling. Unlike other scaling methods, this method does not introduce any additional color into the scaled image.

`resize` scales an image to the desired dimensions using the supplied filter algorithm. Available filters are: `Point, Box, Triangle, Hermite, Hanning, Hamming, Blackman, Gaussian, Quadratic, Cubic, Catrom, Mitchell, Lanczos, Bessel, Sinc`. Most of the filters are FIR (finite impulse response), however, Bessel, Gaussian, and Sinc are IIR (infinite impulse response). Bessel and Sinc are windowed (brought down to zero) with the Blackman filter.

`rotate` creates a new image that is a rotated copy of an existing one. Positive angles rotate counter-clockwise (right-hand rule), while negative angles rotate clockwise. Rotated images are usually larger than the originals and have 'empty' triangular corners. X axis. Empty triangles left over from shearing the image are filled with the background color.

Value

A transformed image in an object of `Image`.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2006-2007

References

ImageMagick: `http://www.imagemagick.org`.

See Also

`Image`

`watershed` *Watershed transformation and watershed based object detection*

Description

Watershed transformation and watershed based object detection.

Usage

```
## S4 method for signature 'Image':
watershed(x, tolerance=1, ext=1, ...)
```

Arguments

x	An object of Image in the Grayscale mode.
tolerance	The minimum height of the object in the units of image intensity between its highest point (seed) and the point where it contacts another object (checked for every contact pixel). If the height is smaller than the tolerance, the object will be combined with one of its neighbors, which is the highest. It is assumed that the function is run on a distance map, therefore the default value is 1. If running the function on an original grayscale image with intensity range [0,1] one should modify this value, down to 0.1 or other which is image specific.
ext	Extension of the neighborhood for the detection of neighboring objects. Higher value smoothes out small objects.
...	Reserved.

Details

The algorithm identifies and separates objects that stand out of the background (zero), in other words to use the water fill, the source image is flipped upside down and the resulting valleys (values with higher intensities) are filled in first until another object or background is met. The deepest valleys (pixels with highest intensity) become indexed first.

Value

An object of Image in the Grayscale with separate objects indexed be positive integers starting from 1. To preview the results visually, use display(display(result)) or use it in combination with paintObjects.

Author(s)

Oleg Sklyar, ⟨osklyar@ebi.ac.uk⟩, 2007

See Also

Image, Image, Image, Image, \code{matchObjects}

Examples

```
## load images
f <- paste( system.file(package="EBImage"), "images/Gene1_G.tif", sep="/" )
ii = readImage(f)
## normalize images
ii = normalize(ii, separate=TRUE)
## segment
mask = thresh(ii, 25, 25, 0.02)
## refine segmentation with morphology filters
mk3 = morphKern(3)
mk5 = morphKern(5)
mask = dilate(erode(closing(mask, mk5), mk3), mk5)
## index objects with 'watershed'
io = watershed( distmap(mask), 1.5, 1)
if (interactive()) display(io)
```

Index

∗Topic **classes**
image,Image-method, 3
Image-class, 6
IndexedImage-class, 8
∗Topic **file**
channel, 4
denoise, 11
display, 13
drawtext, 15
edgeFeatures, 18
floodFill, 27
Image, 9
readImage, 31
∗Topic **manip**
closing, 34
enhance, 17
frameDist, 28
getFeatures, 30
haralickMatrix, 20
hullFeatures, 22
matchObjects, 33
moments, 23
normalize, 35
paintObjects, 36
propagate, 37
resize, 46
rmObjects, 40
segment, 41
stackObjects, 42
thresh, 44
tile, 45
watershed, 47
zernikeMoments, 24
∗Topic **methods**
image,Image-method, 3
Image-class, 6
IndexedImage-class, 8
∗Topic **package**
EBImage-deprecated, 1
EBImage-package, 1
'Image' object creation, copying and assertion, *2*
[,Image-method (*image,Image-method*), 3

affinet (*resize*), 46
affinet,Image-method (*resize*), 46
animate (*display*), 13
animate,array-method (*display*), 13
animate,Image-method (*display*), 13
animate,IndexedImage-method (*display*), 13
Arith (*image,Image-method*), 3
array, *1*, *3*, *4*, *6–8*, *13*, *27*
as.Image (*image,Image-method*), 3
as.Image,array-method (*image,Image-method*), 3
as.Image,IndexedImage-method (*IndexedImage-class*), 8
assert (*Image*), 9
assert,Image,Image-method (*Image*), 9
assert,Image,missing-method (*Image*), 9
athresh (*thresh*), 44
athresh,Image-method (*thresh*), 44

blur (*denoise*), 11
blur,Image-method (*denoise*), 11

cbind, *10*
cgamma (*enhance*), 17
cgamma,Image-method (*enhance*), 17
channel, 4, *27*, *42*
channel,ANY,character-method (*channel*), 4
channel,Image,character-method (*channel*), 4
channelMix (*channel*), 4
character, *4*, *7*
choose.image (*EBImage-deprecated*), 1
chooseImage, *1*
chooseImage (*readImage*), 31
Class 'Image', its accessor method, *2*
Class 'IndexedImage', *2*

closing, 34
closing,Image-method (closing), 34
cmoments (moments), 23
cmoments,IndexedImage,Image-method (moments), 23
cmoments,IndexedImage,missing-method (moments), 23
Color and image color mode conversions, 2
colorMode (Image-class), 6
colorMode<- (Image-class), 6
combine (Image), 9
combine,Image,Image-method (Image), 9
combine,list,missing-method (Image), 9
Combined feature extraction for objects in indexed images, 2
Common generic methods for class 'Image', 2
compression (Image-class), 6
compression<- (Image-class), 6
contrast (enhance), 17
contrast,Image-method (enhance), 17
copy, 6
copy (Image), 9
copy,Image-method (Image), 9
cthresh (thresh), 44
cthresh,Image-method (thresh), 44

denoise, 11
denoise,Image-method (denoise), 11
despeckle (denoise), 11
despeckle,Image-method (denoise), 11
dilate (closing), 34
dilate,Image-method (closing), 34
dim, 4
display, 13, 48
Distance map transform of binary images, 2
distmap, 14
distmap,Image-method (distmap), 14
do.call, 10
drawfont (drawtext), 15
Drawing primitives on images, annotation, 2
drawtext, 15
drawtext,Image,list,list-method (drawtext), 15
drawtext,Image,matrix,character-method (drawtext), 15
drawtext,Image,numeric,character-method (drawtext), 15

EBImage (EBImage-package), 1
EBImage-deprecated, 1
EBImage-package, 1
edge (segment), 41
edge,Image-method (segment), 41
edge.features (EBImage-deprecated), 1
edge.features,IndexedImage-method (EBImage-deprecated), 1
edge.profile (EBImage-deprecated), 1
edge.profile,IndexedImage-method (EBImage-deprecated), 1
edgeFeatures, 1, 18
edgeFeatures,IndexedImage-method (edgeFeatures), 18
edgeProfile, 1
edgeProfile (edgeFeatures), 18
edgeProfile,IndexedImage-method (edgeFeatures), 18
enhance, 17
enhance,Image-method (enhance), 17
Enhancing images and colors, 2
equalize (enhance), 17
equalize,Image-method (enhance), 17
erode (closing), 34
erode,Image-method (closing), 34
Extraction of edge profiles and edge features, 2
Extraction of Haralick texture features and co-occurance matrices (GLCM), 2
Extraction of hull features, 2
Extraction of image moments and moment invariants, 2
Extraction of Zernike moments, 2

features, 36
features (getFeatures), 30
features,IndexedImage-method (getFeatures), 30
fft, 4, 19
fileName, 32
fileName (Image-class), 6
fileName<- (Image-class), 6
fillHull (floodFill), 27

fillHull,IndexedImage-method (floodFill), 27
filter2, 26
filter2,array,matrix-method (filter2), 26
filter2,Image,matrix-method (filter2), 26
filter2-methods (filter2), 26
flip (resize), 46
flip,Image-method (resize), 46
floodFill, 27
floodFill,array-method (floodFill), 27
floor, *15*
flop (resize), 46
flop,Image-method (resize), 46
frameDist, 28
frameDist,Image,Image-method (frameDist), 28
frameDist,Image,missing-method (frameDist), 28

gblur (denoise), 11
gblur,Image-method (denoise), 11
Generate a stack of images for detected objects, *2*
Generate a tiled image from a stack, *2*
getFeatures, 30, *36*, *38*
getFeatures,IndexedImage-method (getFeatures), 30
Grayscale, *10*, *15*, *20*, *23*, *34–36*, *38*, *44*, *48*
Grayscale (Image-class), 6

haralick.features (EBImage-deprecated), 1
haralick.features,IndexedImage,Image-method (EBImage-deprecated), 1
haralick.matrix (EBImage-deprecated), 1
haralick.matrix,IndexedImage,Image-method (EBImage-deprecated), 1
haralickFeatures, *1*, *30*
haralickFeatures (haralickMatrix), 20
haralickFeatures,IndexedImage,Image-method (haralickMatrix), 20
haralickMatrix, *1*, 20
haralickMatrix,IndexedImage,Image-method (haralickMatrix), 20
header (Image), 9
header,Image-method (Image), 9
hist,Image-method (image,Image-method), 3
hull.features (EBImage-deprecated), 1
hull.features,IndexedImage-method (EBImage-deprecated), 1
hullFeatures, *1*, 22, *42*
hullFeatures,IndexedImage-method (hullFeatures), 22

Image, *1*, *3–8*, 9, *9–18*, *23*, *26–28*, *32*, *34–36*, *38*, *41*, *44–48*
Image and color channel thresholding, *2*
Image color manipulation, *2*
Image read/write operations, *2*
Image transformation, rotation, resize etc., *2*
image,Image-method, 3
Image-class, *20*, *25*, *35*
Image-class, 6
imageData, *3*
imageData (Image-class), 6
imageData<- (Image-class), 6
IndexedImage, *13*, *16*, *19–22*, *24–28*, *30*, *31*, *33*, *36–40*, *42*
IndexedImage-class, 8
integer, *4*, *7*, *33*
Interactive image display, *2*
is.Image (Image), 9

list, *7*

Marking detected objects in indexed images, *2*
match, *33*
Matching objects in two indexed images, *2*
matchObjects, 33
matchObjects,IndexedImage,IndexedImage-method (matchObjects), 33
median, *4*
median.Image (image,Image-method), 3
mediansmooth (denoise), 11
mediansmooth,Image-method (denoise), 11
mkball (filter2), 26
mkbox (filter2), 26
modulate (enhance), 17
modulate,Image-method (enhance), 17
moments, *19*, *22*, 23, *25*

moments,Image,missing-method (moments), 23
moments,IndexedImage,Image-method (moments), 23
moments,IndexedImage,missing-method (moments), 23
morphKern (closing), 34
Morphological trasnsformations of binary images, 2

negate (normalize), 35
negate,Image-method (normalize), 35
new, 6, 10
noise (denoise), 11
Noise removal, blurring and smoothing of images, 2
noise,Image-method (denoise), 11
normalize, 35
normalize,Image-method (normalize), 35
normalize2 (normalize), 35
normalize2,Image-method (normalize), 35
numeric, 4, 7

Object removal in indexed images, 2
opening (closing), 34
opening,Image-method (closing), 34

paintObjects, 36, 48
paintObjects,IndexedImage,Image-method (paintObjects), 36
print.Image (image,Image-method), 3
print.IndexedImage (image,Image-method), 3
propagate, 3, 8, 30, 37, 42, 45
propagate,Image,IndexedImage-method (propagate), 37

rbind, 10
read.image (EBImage-deprecated), 1
readImage, 1, 6, 31
reenumerate (rmObjects), 40
reenumerate,IndexedImage-method (rmObjects), 40
resample (resize), 46
resample,Image-method (resize), 46
resize, 46
resize,Image-method (resize), 46
resolution (Image-class), 6
resolution<- (Image-class), 6
rgbImage (channel), 4
rmObjects, 40
rmObjects,IndexedImage,list-method (rmObjects), 40
rmObjects,IndexedImage,numeric-method (rmObjects), 40
rmoments (moments), 23
rmoments,IndexedImage,Image-method (moments), 23
rmoments,IndexedImage,missing-method (moments), 23
rotate (resize), 46
rotate,Image-method (resize), 46

segment, 41
segment,Image-method (segment), 41
Segmentation and edge detection, 2
sharpen (denoise), 11
sharpen,Image-method (denoise), 11
show,Image-method (image,Image-method), 3
smoments (moments), 23
smoments,IndexedImage,Image-method (moments), 23
smoments,IndexedImage,missing-method (moments), 23
sqrt, 4
stackObjects, 24, 29, 42, 46
stackObjects,IndexedImage,Image,character-meth (stackObjects), 42
stackObjects,IndexedImage,Image,list-method (stackObjects), 42
stackObjects,IndexedImage,Image,missing-metho (stackObjects), 42
stackObjects,IndexedImage,Image,numeric-metho (stackObjects), 42
stopIfNotImage (Image), 9

thresh, 44
thresh,Image-method (thresh), 44
tile, 43, 45
tile,Image-method (tile), 45
tile,list-method (tile), 45
TrueColor, 5, 10, 32, 35, 36, 44
TrueColor (Image-class), 6

umask (denoise), 11
umask,Image-method (denoise), 11
untile (tile), 45
untile,Image,numeric-method (tile), 45

Voronoi-based segmentation on
image manifolds, *2*

watershed, *8*, *30*, *36*, *42*, *45*, 47
Watershed transformation and
watershed-based object
detection, *2*
watershed,Image-method
(watershed), 47
write.image *(EBImage-deprecated)*,
1
write.image,Image-method
(EBImage-deprecated), 1
writeImage, *1*
writeImage *(readImage)*, 31
writeImage,Image-method
(readImage), 31

zernike.moments
(EBImage-deprecated), 1
zernike.moments,IndexedImage,Image-method
(EBImage-deprecated), 1
zernikeMoments, *1*, 24, *30*
zernikeMoments,IndexedImage,Image-method
(zernikeMoments), 24