

# Package ‘bioimagetools’

April 11, 2018

**Version** 1.1.3

**Date** 2018-04-11

**Title** Tools for Microscopy Imaging

**Author** Volker Schmid [aut, cre],  
Priyanka Kukreja [ctb],  
Fabian Scheipl [ctb]

**Maintainer** Volker Schmid <stats@volkerschmid.de>

**Depends** R (>= 3.0.0)

**Imports** parallel, tiff, stats, grDevices, utils, EBImage, httr

**Description** Tools for 3D imaging, mostly for biology/microscopy.  
Read and write TIFF stacks. Functions for segmentation, filtering and analysing 3D point patterns.

**License** GPL-3

**URL** <https://www.github.com/bioimaginggroup/bioimagetools>

**RoxygenNote** 6.0.1

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2018-04-11 10:03:23 UTC

## R topics documented:

bwlabel3d . . . . .	2
cmoments3d . . . . .	3
cnnTest . . . . .	3
crossNN . . . . .	4
distance2border . . . . .	5
filterImage3d . . . . .	7
folder.choose . . . . .	8
img . . . . .	8

intensity3D . . . . .	9
K.cross.3D . . . . .	9
L.cross.3D . . . . .	10
mexican.hat.brush . . . . .	11
nearest.neighbour.distribution . . . . .	12
nearestClassDistance . . . . .	13
nearestClassDistances . . . . .	13
outside . . . . .	14
plotNearestClassDistances . . . . .	14
readBMP . . . . .	15
readClassTIF . . . . .	16
readTIF . . . . .	16
segment . . . . .	17
segment.outside . . . . .	18
spots . . . . .	19
standardize . . . . .	20
table.n . . . . .	20
testColoc . . . . .	21
writeTIF . . . . .	22

**Index** **23**

bwlabel3d

*Binary segmentation in 3d*

**Description**

Binary segmentation in 3d

**Usage**

bwlabel3d(img)

**Arguments**

img                    A 3d array. x is considered as a binary image, whose pixels of value 0 are considered as background ones and other pixels as foreground ones.

**Value**

A Grayscale 3d array, containing the labelled version of x.

**Author(s)**

Fabian Scheipl, Volker Schmid

---

cmoments3d	<i>Computes moments from image objects</i>
------------	--

---

**Description**

Computes intensity-weighted centers of objects and their mass (sum of intensities) and size.

**Usage**

```
cmoments3d(mask, ref)
```

**Arguments**

mask	a labeled stack as returned from bwlabel3d
ref	the original image stack

**Value**

a matrix with the moments of the objects in the stack

**Author(s)**

Volker Schmid

---

cnnTest	<i>Permutation Test for cross-type nearest neighbor distances</i>
---------	---

---

**Description**

Permutation Test for cross-type nearest neighbor distances

**Usage**

```
cnnTest(dist, n1, n2, w = rep(1, n1 + n2), B = 999, alternative = "less",
  returnSample = TRUE, parallel = FALSE, ...)
```

**Arguments**

dist	a distance matrix, the upper $n_1 \times n_1$ part contains distances between objects of type 1 the lower $n_2 \times n_2$ part contains distances between objects of type 2
n1	numbers of objects of type 1
n2	numbers of objects of type 2
w	(optional) weights of the objects (length $n_1+n_2$ )
B	number of permutations to generate

alternative	alternative hypothesis ("less" to test H0:Colocalization )
returnSample	return sampled null distribution
parallel	Logical. Should we use parallel computing?
...	additional arguments for mclapply

**Value**

a list with the p.value, the observed weighted mean of the cNN-distances, alternative and (if returnSample) the simulated null dist

**Author(s)**

Fabian Scheipl

---

crossNN	<i>Compute cross-type nearest neighbor distances</i>
---------	--

---

**Description**

Compute cross-type nearest neighbor distances

**Usage**

```
crossNN(dist, n1, n2, w = rep(1, n1 + n2))
```

**Arguments**

dist	a distance matrix, the upper $n1 \times n1$ part contains distances between objects of type 1 the lower $n2 \times n2$ part contains distances between objects of type 2
n1	numbers of objects of type 1
n2	numbers of objects of type 2
w	optional weights of the objects (length $n1+n2$ ), defaults to equal weights

**Value**

a  $(n1+n2) \times 2$  matrix with the cross-type nearest neighbor distances and weights given as the sum of the weights of the involved objects

**Author(s)**

Fabian Scheipl

---

distance2border      *A function to compute the distance from spots to borders of classes*

---

### Description

A function to compute the distance from spots to borders of classes

### Usage

```
distance2border(points, img.classes, x.microns, y.microns, z.microns, class1,
  class2 = NULL, mask = array(TRUE, dim(img.classes)), voxel = FALSE,
  hist = FALSE, main = "Minimal distance to border",
  xlab = "Distance in Microns", xlim = c(-0.3, 0.3), n = 20,
  stats = TRUE, file = NULL, silent = FALSE, parallel = FALSE)
```

### Arguments

points	Data frame containing the coordinates of points in microns as X-, Y-, and Z-variables.
img.classes	3D array (or image) of classes for each voxel.
x.microns	Size of image in x-direction in microns.
y.microns	Size of image in y-direction in microns.
z.microns	Size of image in z-direction in microns.
class1	Which class is the reference class. If is.null(class2), the function computes the distance of points to the border of class (in img.classes).
class2	Which class is the second reference class. If not is.null(class2), the function computes the distance of points from the border between classes class1 and class2. Default: class2=NULL.
mask	Array of mask. Needs to have same dimension as img.classes. Only voxels with mask[i,j,k]==TRUE are used. Default: array(TRUE,dim(img.classes))
voxel	Logical. If TRUE, points coordinates are given as voxels rather than in microns.
hist	Automatically plot histogram using hist() function. Default: FALSE.
main	If (hist) title of histogram. Default: "Minimal distance to border".
xlab	If (hist) description of x axis. Default: "Distance in Microns".
xlim	If (hist) vector of range of x axis (in microns). Default: c(-.3,.3)
n	If (hist) number of bins used in hist(). Default: 20.
stats	If (hist) write statistics into plot. Default: TRUE.
file	If (hist) the file name of the produced png. If NULL, the histogram is plotted to the standard device. Default: NULL.
silent	if TRUE, function remains silent during running time
parallel	Logical. Can we use parallel computing?

## Details

This function computes the distances from points to the border of a class or the border between two classes. For the latter, only points in these two classes are used.

## Value

The function returns a vector with distances. Negative values correspond to points lying in class1.

## Note

Warning: So far no consistency check for arguments is done. E.g., distance2border(randompoints,img.classes=array(1,c(100,100))) will fail with some cryptic error message (because class1 > max(img.classes)).

## Examples

```
## Not run:
#simulate random data
randompoints<-data.frame("X"=runif(100,0,3),"Y"=runif(100,0,3),"Z"=runif(100,0,.5))
# coordinates in microns!
plot(randompoints$X,randompoints$Y,xlim=c(0,3),ylim=c(0,3),pch=19)

# points in a circle
circlepoints<-read.table(system.file("extdata","kreispunkte.table",
                                   package="bioimagertools"),header=TRUE)
plot(circlepoints$X,circlepoints$Y,xlim=c(0,3),ylim=c(0,3),pch=19)

# a circle like image
img<-readTIF(system.file("extdata","kringel.tif",package="bioimagertools"))
img<-array(img,dim(img)) # save as array for easier handling
img(img, z=1)

#and a mask
mask<-readTIF(system.file("extdata","amask.tif",package="bioimagertools"))
img(mask, z=1, col="greyinverted")

xy.microns <- 3 # size in x and y direction (microns)
z.microns <- 0.5 # size in z direction (microns)

# distance from points to class
d1<-distance2border(randompoints, img, xy.microns, xy.microns, z.microns, class1=1,hist=TRUE)
d2<-distance2border(circlepoints, img, xy.microns, xy.microns, z.microns, class1=1,hist=FALSE)
plot(density(d2),type="l")
lines(c(0,0),c(0,10),lty=3)
lines(density(d1),col="blue")

# use mask, should give some small changes
d3<-distance2border(circlepoints, img, xy.microns, xy.microns, z.microns,
                   class1=1,mask=mask,hist=FALSE)

plot(density(d2),type="l")
lines(c(0,0),c(0,10),lty=3)
lines(density(d3),col="blue")
```

```
# distance from border between classes
anotherimg<-img+mask
image(seq(0,3,length=300),seq(0,3,length=300),anotherimg[, ,1])
points(circlepoints,pch=19)
d4<-distance2border(circlepoints, anotherimg, xy.microns, xy.microns, z.microns,
                    class1=1,class2=2)

plot(density(d4),lwd=2)

# this should give the same answer
d5<-distance2border(circlepoints, anotherimg, xy.microns, xy.microns, z.microns,
                    class1=2,class2=1)

lines(density(-d5),lty=3,col="blue",lwd=1.5)

## End(Not run)
```

---

filterImage3d

*Apply filter to 3D images*

---

## Description

A filter is applied to a 3D array representing an image. So far only variance filters are supported.

## Usage

```
filterImage3d(img, filter = "var", window, z.scale = 1, silent = FALSE)
```

## Arguments

<code>img</code>	is a 3d array representing an image.
<code>filter</code>	is the filter to be applied. Options: var: Variance filter.
<code>window</code>	half size of window; i.e. window=1 uses a window of 3 voxels in each direction.
<code>z.scale</code>	ratio of voxel dimension in x/y direction and z direction.
<code>silent</code>	Logical. If FALSE, information on progress will be printed.

## Value

Multi-dimensional array of filtered image data.

---

folder.choose	<i>Choose a folder interactively</i>
---------------	--------------------------------------

---

**Description**

Choose a folder interactively by choosing a file in that folder.

**Usage**

```
folder.choose()
```

**Value**

A character vector of length one giving the folder path.

---

img	<i>Display an image stack</i>
-----	-------------------------------

---

**Description**

Display an image stack

**Usage**

```
img(x, z = NULL, ch = NULL, mask = NULL, col = "grey", low = NULL,
    up = NULL, ...)
```

**Arguments**

x	Image, 2D or 3D Matrix
z	slice to show, default: NULL, expects x to be 2d or 2d+channels
ch	channel. Default: NULL, either only one channel, rgb or channel will be assumed from col
mask	mask for image, voxel outside the mask will be transparent (default: NULL, no mask)
col	Color, either a character ("grey", "greyinvert", "red" ("r"), "green" ("g") or "blue" ("b"), "rgb" for 3D matrices), a vector of character with hex rgb values or a function.
low	minimal value of shown intensity. Default: NULL: use min(x, na.rm=TRUE).
up	maximal value of shown intensity. Default: NULL: use max(x, na.rm=TRUE).
...	other parameters for graphics::image

**Value**

no return



---

intensity3D	<i>Intensity of a 3d Dataset or a Model</i>
-------------	---

---

**Description**

Computing the intensity of a 3d point pattern using kernel smoothing.

**Usage**

```
intensity3D(X, Y, Z, bw = NULL, psz = 25, kernel = "Square")
```

**Arguments**

X	X coordinate
Y	Y coordinate
Z	Z coordinate
bw	bandwidth
psz	pointsize used for discretization (large: fast, but not precise)
kernel	"Square" or "Uniform"

**Value**

3d Array

---

K.cross.3D	<i>K-function cross-type in 3D</i>
------------	------------------------------------

---

**Description**

Calculates an estimate of the cross-type K-function for a multitype point pattern.

**Usage**

```
K.cross.3D(X, Y, Z, X2, Y2, Z2, psz = 25, width = 1, intensity = NULL,
intensity2 = NULL, parallel = FALSE, verbose = FALSE)
```

**Arguments**

X	X coordinate of first observed point pattern in microns.
Y	Y coordinate
Z	Z coordinate
X2	X coordinate of second observed point pattern
Y2	Y coordinate
Z2	Z coordinate
psz	pointsize used for discretization. Smaller values are more precise, but need more computation time.
width	maximum distance
intensity	intensity of first pattern. Only if
	$\lambda(s) \neq \lambda$
	.
intensity2	intensity of second pattern
parallel	Logical. Can we use parallel computing?
verbose	Plot verbose information

**Value**

a list of breaks and counts.

---

L.cross.3D

*L-function cross-type in 3d*

---

**Description**

Calculates an estimate of the cross-type L-function for a multitype point pattern.

**Usage**

```
L.cross.3D(X, Y, Z, X2, Y2, Z2, psz = 25, width = 1, intensity = NULL,
intensity2 = NULL, parallel = FALSE, verbose = FALSE)
```

**Arguments**

X	X coordinate of first observed point pattern in microns.
Y	Y coordinate
Z	Z coordinate
X2	X coordinate of second observed point pattern
Y2	Y coordinate

Z2	Z coordinate
psz	pointsize used for discetization. Smaller values are more precise, but need more computation time.
width	maximum distance
intensity	intensity of first pattern. Only if
	$\lambda(s)! = \lambda$
	.
intensity2	intensity of second pattern
parallel	Logical. Can we use parallel computing?
verbose	Plot verbose information

**Value**

a list of breaks and counts.

---

mexican.hat.brush	<i>Mexican hat brush to use with filter2</i>
-------------------	--

---

**Description**

Mexican hat brush to use with filter2

**Usage**

```
mexican.hat.brush(n = 7, sigma2 = 1)
```

**Arguments**

n	size of brush
sigma2	standard deviation

**Value**

brush

---

nearest.neighbour.distribution

*Nearest neighbour distribution (D curve)*

---

## Description

Nearest neighbour distribution (D curve)

## Usage

```
nearest.neighbour.distribution(X, Y, Z, X2 = X, Y2 = Y, Z2 = Z,  
  same = TRUE, psz = 25, main = "Nearest neighbour distribution",  
  file = NULL, return = FALSE)
```

## Arguments

X	X coordinates of point pattern 1
Y	Y coordinates of point pattern 1
Z	Z coordinates of point pattern 1
X2	X coordinates of point pattern 2
Y2	Y coordinates of point pattern 2
Z2	Z coordinates of point pattern 2
same	binary, FALSE for cross D curve
psz	pointsize for discretization
main	Title for graphic
file	File name for PNG file. If NULL, plots to standard device.
return	Logical. Return histogram?

## Value

histogramm of nearest neighbours

## Examples

```
p<-read.csv(system.file("extdata","cell.csv",package="bioimageroots"))  
nearest.neighbour.distribution(p$X,p$Y,p$Z)
```

---

nearestClassDistance *Title Find distance to next neighbour of a specific class*

---

**Description**

Title Find distance to next neighbour of a specific class

**Usage**

```
nearestClassDistance(coord, img, class, voxelsize, step = 0)
```

**Arguments**

coord	coordinate of relevant voxel
img	image array of classes
class	class to find
voxelsize	vector of length three. size of voxel in X-/Y-/Z-direction
step	size of window to start with

**Value**

distance to nearest voxel of class "class"

---

nearestClassDistances *Find all distances to next neighbour of all classes*

---

**Description**

Find all distances to next neighbour of all classes

**Usage**

```
nearestClassDistances(img, voxelsize = NULL, size = NULL, classes = 7,
  maxdist = NULL, silent = FALSE, cores = 1)
```

**Arguments**

img	Image array of classes
voxelsize	Real size of voxels in microns.
size	Real size of image in microns. Either size or voxelsize must be given.
classes	Number of classes
maxdist	Maximum distance to consider
silent	Remain silent?
cores	Number of cores available for parallel computing

**Value**

array with distances

---

outside

*Segmentation of the background of 3D images based on classes*

---

**Description**

Segmentation of the background of 3D images based on classes

**Usage**

```
outside(img, what, blobsize = 1)
```

**Arguments**

`img` is a 3d array representing an image.

`what` is an integer of the class of the background.

`blobsize` is an integer, representing the minimal diameter for bridges from the outside. E.g., a `blobsize=3` allows for holes of size  $2*(blobsize-1)=4$  in the edge of the object.

**Value**

A binary 3d array: 1 outside the object, 0 inside the object

---

plotNearestClassDistances

*Title Plot nearest class distances*

---

**Description**

Title Plot nearest class distances

**Usage**

```
plotNearestClassDistances(distances, method, classes = length(distances),
  ylim = c(0, 1), qu = 0.01, mfrow = NULL)
```

**Arguments**

distances	list of list with distances as produced by nearestClassDistances()
method	"boxplot", "min" or "quantile"
classes	number of classes, default=7
ylim	limits for distances, default=c(0,1)
qu	quantile for method="quantile"; default 0.01
mfrow	mfrow option forwarded to par; default NULL, computes some optimal values

**Value**

plots

---

readBMP	<i>Read bitmap files</i>
---------	--------------------------

---

**Description**

Read 2D grey-value BMP files

**Usage**

```
readBMP(file)
```

**Arguments**

file            A character vector of file names or URLs.

**Value**

Returns a matrix with BMP data as integer.

**Author(s)**

Volker J. Schmid

**Examples**

```
bi<-readBMP(system.file("extdata/V.bmp",package="bioimageroots"))
image(bi,col=grey(seq(1,0,length=100)))
```

---

readClassTIF	<i>Read TIF file with classes</i>
--------------	-----------------------------------

---

**Description**

Read TIF file with classes

**Usage**

```
readClassTIF(file, n = 7)
```

**Arguments**

file	file
n	number of classes

**Value**

array

---

readTIF	<i>Read tif stacks</i>
---------	------------------------

---

**Description**

Read tif stacks

**Usage**

```
readTIF(file = file.choose(), native = FALSE, as.is = FALSE,  
channels = NULL)
```

**Arguments**

file	Name of the file to read from. Can also be an URL.
native	determines the image representation - if FALSE (the default) then the result is an array, if TRUE then the result is a native raster representation (suitable for plotting).
as.is	attempt to return original values without re-scaling where possible
channels	number of channels

**Value**

3d or 4d array



**Examples**

```
kringel <- readTIF(system.file("extdata", "kringel.tif", package="bioimagetools"))
img(kringel)
```

segment

*Segmentation of 3D images using EM algorithms***Description**

segmentation of 3D images using EM algorithms

**Usage**

```
segment(img, nclust, beta, z.scale = 0, method = "cem", varfixed = TRUE,
  maxit = 30, mask = array(TRUE, dim(img)), priormu = rep(NA, nclust),
  priormusd = rep(NULL, nclust), min.eps = 10^{ -7 },
  inforce.nclust = FALSE, start = NULL, silent = FALSE)
```

**Arguments**

img	is a 3d array representing an image.
nclust	is the number of clusters/classes to be segmented.
beta	is a matrix of size nclust x nclust, representing the prior weight of classes neighbouring each other.
z.scale	ratio of voxel dimension in x/y direction and z direction. Will be multiplied on beta for neighbouring voxel in z direction.
method	only "cem" classification EM algorithm implemented.
varfixed	is a logical variable. If TRUE, variance is equal in each class.
maxit	is the maximum number of iterations.
mask	is a logical array, representing the voxels to be used in the segmentation.
priormu	is a vector with mean of the normal prior of the expected values of all classes. Default is NA, which represents no prior assumption.
priormusd	is a vector with standard deviations of the normal prior of the expected values of all classes.
min.eps	stop criterion. Minimal change in sum of squared estimate of mean in order to stop.
inforce.nclust	if TRUE enforces number of clusters to be nclust. Otherwise classes might be removed during algorithm.
start	?
silent	if TRUE, function remains silent during running time

**Value**

A list with "class": 3d array of class per voxel; "mu" estimated means; "sigma": estimated standard deviations.

**Examples**

```
## Not run:
original<-array(1,c(300,300,50))
for (i in 1:5)original[(i*60)-(0:20),,]<-original[(i*60)-(0:20),,]+1
for (i in 1:10)original[(i*30)-(0:15),,]<-original[(i*30)-(0:15),,]+1
original[,,26:50]<-4-aperm(original[,,26:50],c(2,1,3))

img<-array(rnorm(300*300*50,original,.2),c(300,300,50))
img<-img-min(img)
img<-img/max(img)

try1<-segment(img,3,beta=0.5,z.scale=.3)
print(sum(try1$class!=original)/prod(dim(original)))

beta<-matrix(rep(-.5,9),nrow=3)
beta<-beta+1.5*diag(3)
try2<-segment(img,3,beta,z.scale=.3)
print(sum(try2$class!=original)/prod(dim(original)))

par(mfrow=c(2,2))
img(original)
img(img)
img(try1$class)
img(try2$class)

## End(Not run)
```

---

segment.outside

*Segmentation of the background of 3D images based on automatic threshold*


---

**Description**

Segmentation of the background of 3D images. Starting from the borders of the image, the algorithm tries to find the edges of an object in the middle of the image. From this, a threshold for the edge is defined automatically. The function then return the a logical array representing voxel inside the object.

**Usage**

```
segment.outside(img, blobsize = 1)
```

**Arguments**

`img` is a 3D array representing an image.

`blobsize` is an integer, representing the minimal diameter for bridges from the outside. E.g., a `blobsize=3` allows for holes of size  $2*(blobsize-1)=4$  in the edge of the object.

**Value**

A binary 3D array: 1 outside the object, 0 inside the object.

**Examples**

```
kringel <- readTIF(system.file("extdata", "kringel.tif", package="bioimagerools"))
out <- segment.outside(kringel)
img(out, z=1)
```

---

spots

*Find spots based on threshold and minimum total intensity*

---

**Description**

Find spots based on threshold and minimum total intensity

**Usage**

```
spots(img, mask, thresh.offset = 0.1, min.sum.intensity = 0, zero = NA,
      max.spots = NULL, return = "full")
```

**Arguments**

`img` image array

`mask` mask

`thresh.offset` threshold for minimum voxel intensity

`min.sum.intensity` threshold for minimum total spot intensity

`zero` if NA, background is set to NA, if 0, background is set to 0

`max.spots` find max.spots spots with highest total intensity

`return` "mask" returns binarized mask, "intensity" returns intensity for spots, zero or NA otherwise "label" return labeled (numbered) spots

**Value**

array

---

standardize	<i>Standardize images</i>
-------------	---------------------------

---

**Description**

Standardizes images in order to compare different images. Mean of standardized image is 0.5, standard deviation is sd.

**Usage**

```
standardize(img, mask = array(TRUE, dim(img)), log = FALSE, N = 32,
            sd = 1/6)
```

**Arguments**

img	is a 2d/3d array representing an image.
mask	a mask.
log	Logical. Transform to log scale before standardization?
N	number of classes.
sd	standard deviation.

**Value**

Multi-dimensional array of standardized image.

**Examples**

```
#simuliere Daten zum Testen
test2<-runif(128*128,0,1)
test2<-sort(test2)
test2<-array(test2,c(128,128))
img(test2)
# Standardisiere test2 in 32 Klassen
std<-standardize(test2,N=32,sd=4)
```

---

table.n	<i>Cross Tabulation and Table Creation (including empty classes)</i>
---------	--

---

**Description**

Cross Tabulation and Table Creation (including empty classes)

**Usage**

```
table.n(x, m = max(x, na.rm = TRUE), percentage = FALSE, weight = NULL,
        parallel = FALSE)
```

**Arguments**

x	R object with classes
m	maximum number of classes
percentage	boolean. If TRUE result is in percentages.
weight	weight for each voxel
parallel	Logical. Can we use parallel computing?

**Value**

vector with (weighted) counts (including empty classes)

**Author(s)**

Volker Schmid 2013-2016

**Examples**

```
x <- c(1,1,2,2,4,4,4)
table.n(x)
# [1] 2 2 0 3
table.n(x, m=5)
# [1] 2 2 0 3 0
table.n(x, weight=c(1,1,1,2,.5,.5,.5))
# [1] 2.0 3.0 0.0 1.5
```

---

testColoc

*Permutation Test for cross-type nearest neighbor distances*


---

**Description**

Permutation Test for cross-type nearest neighbor distances

**Usage**

```
testColoc(im1, im2, hres = 0.102381, vres = 0.25, B = 999,
  alternative = "less", returnSample = TRUE, ...)
```

**Arguments**

im1	image stack as returned by preprocessing
im2	image stack as returned by preprocessing
hres	horizontal resolution of the stacks
vres	vertical resolution of the stacks
B	number of permutations to generate

alternative	alternative hypothesis ("less" to test H0:Colocalization )
returnSample	return sampled null distribution
...	additional arguments for papply

**Value**

a list with the p.value, the observed weighted mean of the cNN-distances

**Author(s)**

Fabian Scheipl

---

writeTIF	<i>Writes image stack into a TIFF file. Wrapper for writeTIFF</i>
----------	---

---

**Description**

Writes image stack into a TIFF file. Wrapper for writeTIFF

**Usage**

```
writeTIF(img, file, bps = attributes(img)$bits.per.sample, twod = FALSE,
         reduce = TRUE, attr = attributes(img), compression = "none")
```

**Arguments**

img	An image, a 3d or 4d array.
file	File name.
bps	number of bits per sample (numeric scalar). Supported values in this version are 8, 16, and 32.
twod	Dimension of channels. TRUE for 2d images, FALSE for 3d images.
reduce	if TRUE then writeTIFF will attempt to reduce the number of planes in native rasters by analyzing the image to choose one of RGBA, RGB, GA or G formats, whichever uses the least planes without any loss. Otherwise the image is always saved with four planes (RGBA).
attr	Attributes of image stack. Will be propagated to each 2d image.
compression	(see ?writeTIFF)

# Index

`bwlabel3d`, 2

`cmoments3d`, 3  
`cnnTest`, 3  
`crossNN`, 4

`distance2border`, 5

`filterImage3d`, 7  
`folder.choose`, 8

`img`, 8  
`intensity3D`, 9

`K.cross.3D`, 9

`L.cross.3D`, 10

`mexican.hat.brush`, 11

`nearest.neighbour.distribution`, 12  
`nearestClassDistance`, 13  
`nearestClassDistances`, 13

`outside`, 14

`plotNearestClassDistances`, 14

`readBMP`, 15  
`readClassTIF`, 16  
`readTIF`, 16

`segment`, 17  
`segment.outside`, 18  
`spots`, 19  
`standardize`, 20

`table.n`, 20  
`testColoc`, 21

`writeTIF`, 22