Package 'RCI'

March 25, 2013

Title R Calcium Imaging Analysis

Version 1.0

Date 2010-01-09

Author Bronwyn Woods

Maintainer Bronwyn WOods <bwoods@cmu.edu>

Description Tools for analyzing in-vivo two-photon calcium imaging data.

License GPL

Imports RSQLite, randomForest, gWidgets, gWidgetsRGtk2, cairoDevice,R.utils, mgcv, RSEIS

Collate 'datainport.R' 'motion.R' 'plotting.R' 'segmentation.R'
'gui.R' 'imageprocessing.R' 'timeseries.R' 'clustering.R'

Archs i386, x86_64

R topics documented:

`AddConMat` *INTERNAL Adds the overlap edges to a mask database*

Description

INTERNAL Adds the overlap edges to a mask database

Usage

```
AddConMat(db, cmat, ids)
```

Arguments

db	the mask database object
cmat	a matrix of 0/1 values giving the locations of edges between masks
ids	a vector giving the ids of the masks in cmat (in order)

Value

NULL

AddMask *Plots a mask over an already plotted image*

Description

Plots a mask over an already plotted image

Usage

```
AddMask(mask, rgb = runif(3), alpha = 0.5, ...)
```

Arguments

mask	the specification of the mask
rgb	a vector of length 3 giving the color of the mask in RGB (defaults to random)
alpha	the alpha transparency value of the mask (between 0 and 1)
...	additional graphing parameters

Details

Given a mask as either a matrix of logicals or a matrix with 1's on the mask, over-plot a semi-transparent colored region on an already plotted image.

Value

NULL

AddMasks — *Generate masks according to the given method and add them to the database*

Description

Generate masks according to the given method and add them to the database

Usage

```
AddMasks(db, calexp, method)
```

Arguments

db	the mask database object
calexp	the calcium experiment data object
method	what method should be used to generate masks to add

AddMaskSet — *Plots sets of masks over an already plotted image*

Description

Plots sets of masks over an already plotted image

Usage

```
AddMaskSet(mask, alpha = 0.5, ...)
```

Arguments

mask	the specification of the mask, unique values for each mask set, and 0 or NA in background
alpha	the alpha transparency value of the mask (between 0 and 1)
...	additional graphing parameters

Details

Given a matrix with unique integers for each mask set, overplot each mask set in a different color (randomly chosen)

Value

NULL

`AssignToPeaks` *Assigns the non-zero pixels of 'region' to one of the maxima of the iamge by hillclimbing on image*

Description

Assigns the non-zero pixels of 'region' to one of the maxima of the iamge by hillclimbing on image

Usage

```
AssignToPeaks(region, image, restrict = T)
```

Arguments

`region`	a matrix with 1 in the regions to be assigned and 0 elsewhere
`image`	the image matrix
`restrict`	boolean. should the hill-climbing be restricted to a path entirely within region

Value

a matrix with unique integers in the pixels of region corresponding to each local maxima

`ClipImage` *Clips a border from around an image matrix*

Description

Clips a border from around an image matrix

Usage

```
ClipImage(image, border = NULL, size = NULL)
```

Arguments

`image`	the image matrix to clip
`border`	the size of the border to clip. Must be less than half the image size
`size`	the resulting size of the image.

Details

uses the size argument if given, else uses the border argument, else returns the original image

Value

a matrix with the center (nrow-2*border) by (ncol-2*border) pixels of the image

ClusterCells *Clustering*

Description

Clustering

Usage

```
ClusterCells(calexp, mask, k, criteria = "cor",
  freq = c(0.78, 0.81), dt = 0.1247232)
```

Arguments

calexp	the calexp object with the data
mask	a mask identifying the cells to be clustered. Each unique non-zero/NA value in the mask indicates a cell to be clustered.
k	the number of clusters to find
criteria	the criteria to use for clustering – 'cor' (correlation) 'phase' (phase of frequency specified in freq)
freq	the frequency band to use to extract the phase for phase-clustering

ClusterCorrelation *See correlation with clusters*

Description

See correlation with clusters

Usage

```
ClusterCorrelation(calexp, clusters)
```

Arguments

calexp	the calexp object
clusters	the cluster object as returned from ClusterCells

CompMaskC *INTERNAL Compute the overlap of a single mask with a list of masks*

Description

INTERNAL Compute the overlap of a single mask with a list of masks

Usage

```
CompMaskC(mask, masklist)
```

Arguments

mask	A single mask formatted as a vector of mask indices with or without the negative id as the first element of the vector.
masklist	A list of sparse masks as returned by GetMasks. Each element of this list is a vector whose first element is the negative id of the mask and whose other elements are the sorted indices of the mask pixels.

Details

Computes the overlap matrix between a mask and a list of other masks using C code for efficiency

Value

a vector whose elements are 0 or 1 giving the overlap relationships between the masks. The values are sorted in the vector in the same order as they are given in masklist.

CompMasksC *INTERNAL Compute the overlap matrix between a set of masks*

Description

INTERNAL Compute the overlap matrix between a set of masks

Usage

```
CompMasksC(masklist)
```

Arguments

masklist	A list of sparse masks as returned by GetMasks. Each element of this list is a vector whose first element is the negative id of the mask and whose other elements are the sorted indices of the mask pixels.

Details

Computes the overlap matrix between a set of masks using C code for efficiency.

Value

a matrix whose elements are 0 or 1 giving the overlap relationships between the masks. The masks are sorted in the matrix in the same order as they are given in masklist.

ConMaskDb *Connects to an experiment's mask database*

Description

Connects to an experiment's mask database

Usage

```
ConMaskDb(path)
```

Arguments

path	the path to the SQLite database to connect to

Value

a connection object as returned by dbConnect in the DBI package

---

ConvolveImage *Convolves an image with the given kernel matrix*

Description

Convolves an image with the given kernel matrix

Usage

```
ConvolveImage(image, kernel, circular = T)
```

Arguments

image	a matrix with the image
kernel	a matrix with the kernel (should be smaller than the image)
circular	boolean, should the convolution be circular (default) or should the image be padded with zeros to prevent circular convolution

Details

Uses Fourier methods to convolve the given image with the given kernel

Value

a matrix of the same size as image with the convolved image

`CountHolesC` *INTERNAL Counts the number of pixels not in a mask that are surrounded by at least 3 mask pixels*

Description

INTERNAL Counts the number of pixels not in a mask that are surrounded by at least 3 mask pixels

Usage

```
CountHolesC(mask)
```

Arguments

`mask`	the mask in which to count holes. NA or 0 in the background.

Details

Uses C code from the file countholesC.c

Value

an integer giving the number of holes in the mask

`CreateCalExpFromCSV` *Convert a folder of text images to a calexp data object*

Description

Convert a folder of text images to a calexp data object

Usage

```
CreateCalExpFromCSV(name, imgdir, nchans = 2)
```

Arguments

`name`	a short name to identify this experiment
`imgdir`	a string giving the directory path for the directory containing the csv images
`nchans`	the number of channels that exist in the data

Details

This function Converts a directory of csv text files into a calexp data object in R. Assumes that the images are individual csv text files and that they are alphabetically in order by channel and then by time index. The directory must contain only these csv image files. Each image must have the same dimensions, and there must be the same number of images for each channel.

Value

an object of class calexp

name	the name passed in as an argument to this function
data	an array containing the image data, with dimensions nchans-nrows-ncols

---

CreateCalExpFromText

Convert a folder of text images to a calexp data object

Description

Convert a folder of text images to a calexp data object

Usage

```
CreateCalExpFromText(name, imgdir, nchans = 2)
```

Arguments

name	a short name to identify this experiment
imgdir	a string giving the directory path for the directory containing the csv images
nchans	the number of channels that exist in the data

Details

This function Converts a directory of text files into a calexp data object in R. Assumes that the images are individual text files and that they are alphabetically in order by channel and then by time index. The directory must contain only these image files. Each image must have the same dimensions, and there must be the same number of images for each channel.

Value

an object of class calexp

name	the name passed in as an argument to this function
data	an array containing the image data, with dimensions nchans-nrows-ncols

`CreateCurExp` *INTERNAL Create an object to store information about the currently selected experiment.*

Description

INTERNAL Create an object to store information about the currently selected experiment.

Usage

```
CreateCurExp()
```

Details

Creates a list that stores information about the currently selected experiment.

Value

A list with fields to store information about the experiment (initially empty)

`name`	the experiment name
`data`	if loaded, the data object for this object
`db`	the database connection for this experiment
`nmasks`	the number of candidate masks in the database for this experiment
`features`	the tags of the features that exist in this database
`sources`	the tage for the mask sources present in this database
`selmat`	a matrix where the first column is the ID of the mask and the second column gives the annotation for the mask
`sms`	the list of sparse masks for the experiment retreived from the database with GetMasks()
`mimg1`	the matrix giving the mean image for channel 1
`mimg2`	the matrix giving the mean image for channel 2
`nx`	the number of columns in the images for this experiment
`ny`	the number of rows in the images

`CreateDbController` *INTERNAL Creates an empty database controller*

Description

INTERNAL Creates an empty database controller

Usage

```
CreateDbController()
```

Details

A database controller holds information about the directories where the databases, data, classifiers, and helper files are stored.

Value

A list

`db.directory` the directory holding SQLite databases

`data.directory`
: the directory holding data associated with each database. each of these should have a \$data element

`helper.directory`
: the directory in which to place helper files generated by the GUI

`classifier.direcoty`
: the directory that contains the classifiers used in segmentation

`expdf` a data.frame in which to put information about each experiment, currently empty

---

DbAddMask *Add a mask to a database*

Description

Add a mask to a database

Usage

```
DbAddMask(db, mask, source)
```

Arguments

`db`	a database connection object
`mask`	a matrix giving the mask to add to the database (T/F, 0/1, or NA/1)
`source`	a string giving the tag for the source of the mask

Details

Adds the given mask to the database. If the mask is already in the database, increments the count for the source of the mask (or adds a new count for a new source)

Value

NULL

`DbSetup` *INTERNAL Creates an empty mask database with the appropriate tables*

Description

INTERNAL Creates an empty mask database with the appropriate tables

Usage

```
DbSetup(db)
```

Arguments

`db`	the database object for which to create the mask tables

Value

NULL

`EmbedAndTaperImage` *Embeds an image in a larger matrix of 0's and tapers the image edges using a Hanning window*

Description

Embeds an image in a larger matrix of 0's and tapers the image edges using a Hanning window

Usage

```
EmbedAndTaperImage(img, taperamt, size = NULL,
  border = NULL)
```

Arguments

`img`	the image to embed and taper
`taperamt`	the width of the taper on the edges of the image. Must be less than or equal to half the image width
`border`	the width of the border of 0's to add

Details

uses size if given, else uses border, else doesn't embed

Value

an image that has been embedded and tapered

`EmbedImage` *Embeds an image matrix in a larger matrix with a border of 0's*

Description

Embeds an image matrix in a larger matrix with a border of 0's

Usage

```
EmbedImage(image, border = NULL, size = NULL)
```

Arguments

`image`	the image to embed
`border`	the width of the border to add around the edges
`size`	the resulting size of the image - this must be bigger than the dimensions of image

Details

uses the size argument if given, else uses the border argument, else returns the original image

Value

a matrix of size (nrow+2*border) by (ncol+ 2*border)

`FFTPhaseCor` *INTERNAL Computes sub-pixel shifts values using phase correlation (FFT implementation)*

Description

INTERNAL Computes sub-pixel shifts values using phase correlation (FFT implementation)

Usage

```
FFTPhaseCor(img1, img2, upsamp = 2, taper = TRUE,
  cortaper = TRUE, subpixel = "gauss", subrad = 3)
```

Arguments

`img1`	matrix giving the first image (the reference)
`img2`	matrix giving the second image (to be shifted)
`upsamp`	the factor by which the fft matrix should be expanded
`taper`	boolean, should the images be tapered before alignment
`cortaper`	boolean, should the normalized cross-spectrum be tapered before being (inverse) transformed
`subpixel`	'none' for no additional subpixel fitting, 'gauss' for Gaussian fit, 'poc' for poc function fitting
`subrad`	the radius of the submatrix used to compute the subpixel fits

Details

Computes the sub-pixel shifts by computing the upsampled phase correlation between the two images and finding the maximum. If the parameter gausfit is TRUE, then a gaussian is fit around the peak of the phase correlation function to get additional sub-pixel shift information. This is on top of any upsampling

Value

a vector of length 2 giving the magnitude of the estimted x and y shift returns NA in the case of improper input

---

FFTXCor *INTERNAL Computes sub-pixel shifts values using FFT*

Description

INTERNAL Computes sub-pixel shifts values using FFT

Usage

```
FFTXCor(img1, img2, upsamp = 1, taper = 0)
```

Arguments

img1	matrix giving the first image (the reference)
img2	matrix giving the second image (to be shifted)
upsamp	the factor by which the fft matrix should be expanded
taper	number of pixels to taper the data on the edges of the image

Details

Computes the sub-pixel shifts by computing the upsampled cross-correlation between the two images and finding the maximum. Computes the upsampled cross-correlation by embedding the product of FT(img1)* and FFT(img2) in a larger matrix of 0's determined by the upsampling factor.

Value

a vector of length 2 giving the magnitude of the estimted x and y shift returns NA in the case of improper input

`FilterVector` *INTERNAL Filters a vector by frequency using a butterworth filter*

Description

INTERNAL Filters a vector by frequency using a butterworth filter

Usage

```
FilterVector(vec, low, high, order = 8, dt = 1/1000,
  type = "BP")
```

Arguments

`vec`	the vector to filter
`low`	the lower value of the filter
`high`	the higher value of the filter
`order`	the order of the butterworth filter
`dt`	the time (in seconds) of one datapoint. 1/frequency in hz
`type`	the type of filter, defaults to "BP" bandpass filter. Can also choose other filters offered by the butfilt function

Value

the filtered vector

`GetDataFeatures` *INTERNAL Computes the features related to the data under a mask, adding them to the database*

Description

INTERNAL Computes the features related to the data under a mask, adding them to the database

Usage

```
GetDataFeatures(db, data, cormat)
```

Arguments

`db`	a database connection
`data`	the data array for this experiment
`cormat`	the pixel-pixel corrlations for channel 2 for this data

Details

Computes features of all masks in the database or a list of masks specified by id. The features computed are currently:
var1 - the variance of the pixel means for channel 1
var2 - the variance of the pixel means for channel 2
var1eq - the variance of the pixel means for the equalized version of channel 1
var2eq - the variance of the pixel means for the equalized version of channel 2
mean1eq - the mean of the pixel means for the equalized version of channel 1
mean2eq - the mean of the pixel means for the equalized version of channel 2
cor2 - the mean pixel-pixel correlation between the map pixels in channel 2
cor2min - the min pixel-pixel correlation between the map pixels in channel 2
cor2max - the max pixel-pixel correlation between the map pixels in channel 2

Value

NULL

GetExtrema *Finds the extrema in an image.*

Description

Finds the extrema in an image.

Usage

```
GetExtrema(image, maxima = T)
```

Arguments

image	the image matrix
maxima	boolean, should this function find maxima (default). If false, finds minima

Value

a matrix with 1 at maxima (or minima) and 0 elsewhere

GetInnerMasks *INTERNAL Selects the masks from the given list that are contained in a region*

Description

INTERNAL Selects the masks from the given list that are contained in a region

Usage

```
GetInnerMasks(framemat, masklist)
```

Arguments

framemat	a matrix of the same size as the masks in masklist with non-NA pixels specifying the region in which to find masks
masklist	a list, as returned by GetMasks, of sparse masks (vectors where the first element is the negative id of the mask and the other elements are the mask indices)

Details

Given a list of masks as returned by GetMasks and a matrix with a mask specifying a region, returns the masks in the masklist that are completely contained in the given region.

GetMask *Return the requested mask from the specified database*

Description

Return the requested mask from the specified database

Usage

```
GetMask(db, id, format = "sparse")
```

Arguments

db	a database connection
id	the id of the mask to return
format	"sparse" for a sparse mask in vector form, "matrix" for a matrix mask

Value

either a vector giving the indices of the requested mask or a matrix version of the mask

---

GetMasks *Returns a list of the masks in a database*

Description

Returns a list of the masks in a database

Usage

```
GetMasks(db)
```

Arguments

db	a database connection

Value

a list of vectors, each vector specifying a mask. The first element of each mask vector is the negative index of the mask. The remaining elements of each vector are the indices of the mask pixels.

GetSeries — *Return average time series for each cell in a mask*

Description

Return average time series for each cell in a mask

Usage

```
GetSeries(calexp, mask, channel = 2)
```

Arguments

calexp	the calexp with the data
mask	the mask identifying cells. Each unique non-zero/NA value in the mask indicates a cell to be clustered.
channel	the channel to get the cell traces from

GetShapeFeatures — *INTERNAL Computes the features related to just the shape of masks, adding them to the database*

Description

INTERNAL Computes the features related to just the shape of masks, adding them to the database

Usage

```
GetShapeFeatures(db, mids = NULL)
```

Arguments

db	a database connection
mids	an optional vector of the mask ids for which to extract features

Details

Computes features of all masks in the database or a list of masks specified by id. The features computed are currently:
npixels - the number of pixels in a mask
nholes - the number of non-mask pixels that are surrounded by at least 3 mask pixels
bboxratio - the ratio of the area of the mask's bounding box and the number of pixels
in the mask hullratio - the ratio of the area of the mask's convex hull and the number of pixels in the mask

Value

NULL

`HillClimbC` *INTERNAL Perform hill climbing on a matrix starting from a given point and returning the local maxima that is reached.*

Description

INTERNAL Perform hill climbing on a matrix starting from a given point and returning the local maxima that is reached.

Usage

```
HillClimbC(y, x, img)
```

Arguments

`y`	Starting row
`x`	Starting column
`img`	The matrix on which to performt the hillclimbing

Details

Uses C code in hillclimbC.c

Value

a vector of 2 numbers giving the coordinates of the peak found by hillclimbing

`HistEqualC` *INTERNAL Computed the histogram equalization of a matrix.*

Description

INTERNAL Computed the histogram equalization of a matrix.

Usage

```
HistEqualC(mat, fullmax = 4096)
```

Arguments

`mat`	the matrix to equalize
`fullmax`	the range to equalize to

Details

Uses C code in histequalC.c

Value

the equalized matrix

Image — *Plots an image of the given matrix with the origin in the upper left*

Description

Plots an image of the given matrix with the origin in the upper left

Usage

```
Image(img, col = grey(seq(0, 1, 0.001)), ...)
```

Arguments

img	the image matrix to plot
col	a list of colors to use for plotting, defaults to grey
...	additional graphing parameters

Value

NULL

ImageToCoordMat — *INTERNAL Converts an image matrix to a matrix with coordinates and values in the columns*

Description

INTERNAL Converts an image matrix to a matrix with coordinates and values in the columns

Usage

```
ImageToCoordMat(img)
```

Arguments

img	the matrix to convert

Value

A matrix of size npixels-by-3. The first coordinate is the row, the the column and the third the intensity.

`IntensityCorrection` *Performs intensity correction on the given calcium experiment*

Description

Performs intensity correction on the given calcium experiment

Usage

```
IntensityCorrection(calexp, cortype = "ar", order = 25,
  naclip = T)
```

Arguments

`calexp`	the data to be corrected is in the $data element of this calexp object
`cortype`	the type of correction to perform. 'ar' for autoregressive filter
`order`	the order of the model to fit (for ar type)
`naclip`	should NAs produced at the beginning of the experiment be clipped off (by AR model, for instance)

`InvertMask` *INTERNAL Inverts a mask matrix so that the mask region is turned to backgroun and vice versa*

Description

INTERNAL Inverts a mask matrix so that the mask region is turned to backgroun and vice versa

Usage

```
InvertMask(mask)
```

Arguments

`mask`	the mask matrix to invert, with NA in the background

Value

a matrix with the inverted mask

`LoGKernel` *Returns a Laplacian of Gaussian kernel*

Description

Returns a Laplacian of Gaussian kernel

Usage

```
LoGKernel(kdim, sigma)
```

Arguments

`kdim`	the dimension of the (square) kernel to generate
`sigma`	the standard deviation of the gaussian smoother

Value

a matrix giving the LoG kernel

`MaskDbSetup` *INTERNAL Creates an empty mask database with the appropriate tables*

Description

INTERNAL Creates an empty mask database with the appropriate tables

Usage

```
MaskDbSetup(db)
```

Arguments

`db`	the database object for which to create the mask tables

Value

NULL

`MaskHull` *INTERNAL Computes the convex hull of a mask*

Description

INTERNAL Computes the convex hull of a mask

Usage

```
MaskHull(mask)
```

Arguments

`mask`	the mask for which to find the convex hull. Background pixels should be NA

Details

FIXME: there's the issue that maphull(maphull(x))!=maphull(x), but using this anyway

Value

a matrix with 1's on the convex hull of the mask and NA in the background

`MatrixToSparse` *INTERNAL Converts a matrix mask into a sparse mask. Assumes that the non-mask pixels of the matrix are NA.*

Description

INTERNAL Converts a matrix mask into a sparse mask. Assumes that the non-mask pixels of the matrix are NA.

Usage

```
MatrixToSparse(mat)
```

Arguments

`mat`	The mask as a matrix with NA in non-mask pixels

Value

a vector of indices of the mask pixels

`MultiTaperSpectrum` *Uses multi-taper methods to etimate a spectrum for the given vector*

Description

Uses multi-taper methods to etimate a spectrum for the given vector

Usage

```
MultiTaperSpectrum(vec, dt = 0.1247232, dif = T)
```

`OptimRotate` *INTERNAL Uses optimization of an objective function to compute the best alignment rotation between two images*

Description

INTERNAL Uses optimization of an objective function to compute the best alignment rotation between two images

Usage

```
OptimRotate(img1, img2, taper = TRUE, error = "mse",
  searchrange = c(-0.1, 0.1))
```

Arguments

`img1`	the reference image
`img2`	the image to align
`taper`	should the images be tapered before the rotation is computed (hanning window)
`error`	objective function to be used - "mse" mean squared error, "mae" mean absolute error, "cor" correlation
`searchrange`	the range of rotations to search over in the optimization

Value

a real valued estimate of the optimal alignment rotation

`OptimShift` *INTERNAL Computes the rigid body motion alignment parameters by optimizing some error function comparing the two images. (uses optimization routines in the neldermead package)*

Description

INTERNAL Computes the rigid body motion alignment parameters by optimizing some error function comparing the two images. (uses optimization routines in the neldermead package)

Usage

```
OptimShift(img1, img2, taper = TRUE, error = "mse",
  startval = c(0.1, 0.1, 0), pocstart = TRUE,
  bigsize = NULL)
```

Arguments

`img1`	the reference image
`img2`	the image to align
`taper`	boolean, should the images be tapered before aligning
`bigsize`	the size of the array in which to embed the tapered images (defaults to next power of 2)
`error`	the error function to use. Can be "mse" for mean squared error, mae" for mean absolute error, or "cor" for correlation.
`startval`	a length 3 vector giving the inital values for the optimization (xshift, yshift, theta)
`pocstart`	should the POC method be used to initialize the start values

Value

a vector of length 3 giving the translation and rotation estimates

`OptimTranslate` *Uses optimization of an objective function to compute the best alignment translation between two images (uses optimization routines in the neldermead package)*

Description

Uses optimization of an objective function to compute the best alignment translation between two images (uses optimization routines in the neldermead package)

Usage

```
OptimTranslate(img1, img2, taper = TRUE, error = "mse",
  startval = c(0.1, 0.1), bigsize = NULL)
```

Arguments

img1	the reference image
img2	the image to align
taper	should the images be tapered before the rotation is computed (hanning window)
error	objective function to be used - "mse" mean squared error, "mae" mean absolute error, "cor" correlation
startval	the inital estimate of the shift parameters

Value

a real valued vector of length 2, giving estimates of x and y translation

---

RegisterCalExp *Removes in-plane motion effects using rigid body alignment of the image frames*

Description

Removes in-plane motion effects using rigid body alignment of the image frames

Usage

```
RegisterCalExp(calexp, refimg, channel = 1,
  bigsize = c(256, 256))
```

Arguments

calexp	a calexp object with a \$data field
refimg	a reference image to use for alignment. Should be the same size as the images in calexp\$data
channel	the channel to use for alignment (typically the structual channel)
upsamp	the upsampling factor (this gives the sup-pixel precision of 1/upsamp)

Details

Registers the images in a calexp object by rigid body image alignment of the images in a particular channel to the reference image given. Initial translation parameters are estimated using Phase-Only correlation. The parameters are then optimized using Nelder-Mead optimization of the mean squared error between the images.

Value

a calexp object with a \$registration field. The \$data in the returned object has been registered. The \$registration field records the details of the estimated shifts.

refimg	the reference image used
mpars	the estimated shifts. This is a matrix of size nframes-by-2

RemoveMask *Remove a mask from a mask database*

Description

Remove a mask from a mask database

Usage

```
RemoveMask(db, maskid)
```

Arguments

db	a database connection
maskid	the ID of the mask to remove

Details

Removes a mask, as well as any associated features and edges

Value

NULL

ReorderFFT *INTERNAL Reorders the matrix returned by fft*

Description

INTERNAL Reorders the matrix returned by fft

Usage

```
ReorderFFT(mat, inverse = F)
```

Arguments

mat	a matrix of values to reorder
inverse	if true, takes reordered matrix and returns to order expected by fft. if false, takes matrix from fft and reorders it

Details

Reorders the matrix returned by the R function fft. The R function returns the coefficients from low-to-high-to-low frequencies in both dimensions. The reordering puts the low frequencies in the center of the matrix so that the coefficients go from high-to-low-to-high in each dimension

Value

the reordered matrix

`RotateFFT` *Rotates an image by the given angle using a sequence of Fourier domain shears as described in Eddy 1996.*

Description

Rotates an image by the given angle using a sequence of Fourier domain shears as described in Eddy 1996.

Usage

```
RotateFFT(img, theta, fdomain = FALSE)
```

Arguments

`img`	the image to rotate
`theta`	the angle to rotate the image
`fdomain`	is the image given already in the Fourier domain? It will be returned in the same domain as given (passing in the Fourier domain is helpful to reduce superfluous transforms if performing additional operations in the Fourier domain).

`RotateImg` *INTERNAL Rotates an image by a given number of integer rows and columns*

Description

INTERNAL Rotates an image by a given number of integer rows and columns

Usage

```
RotateImg(mat, x, y)
```

Arguments

`mat`	the matrix to rotate
`x`	the number of columns to rotate
`y`	the number of rows to rotate

Value

the rotated matrix

SetMaskLabel *Sets the label field for a particular mask in a mask database*

Description

Sets the label field for a particular mask in a mask database

Usage

```
SetMaskLabel(db, id, label)
```

Arguments

db	a database connection
id	the id of the mask to label
label	the label to assign to the mask (0=unknown, 1=cell, 2=not cell)

Value

NULL

ShiftFFT *Shifts an image by the given amount, both translation and rotation*

Description

Shifts an image by the given amount, both translation and rotation

Usage

```
ShiftFFT(img, pars, fdomain = FALSE, rotatefirst = FALSE)
```

Arguments

img	the image to shift
pars	a length-3 vector giving (x-translation, y-translation, rotation angle)
fdomain	is the image given in the Fourier domain? It will be returned in the same domain as given (passing in the Fourier domain is helpful to reduce superfluous transforms if performing additional operations in the Fourier domain).
rotatefirst	should rotation be performed before translation

Details

Uses RotateFFT and TranslateFFT to compute result

Value

the shifted image

`ShiftFFTVector` *INTERNAL Shifts a vector by the specified amount using FFT phase shift, but assuming the Fourier transform has already been performed.*

Description

INTERNAL Shifts a vector by the specified amount using FFT phase shift, but assuming the Fourier transform has already been performed.

Usage

```
ShiftFFTVector(vec, amt)
```

Arguments

`vec`	the vector to shift
`amt`	the amount to shift

Value

the circularly shifted vector

`ShiftVector` *INTERNAL Shifts a vector by the specified amount using FFT*

Description

INTERNAL Shifts a vector by the specified amount using FFT

Usage

```
ShiftVector(vec, amt)
```

Arguments

`vec`	the vector to shift
`amt`	the amount to shift

Value

the circularly shifted vector

SimpleModesC *INTERNAL Finds the local maxima in an image*

Description

INTERNAL Finds the local maxima in an image

Usage

```
SimpleModesC(img, min = 0)
```

Arguments

img	the image in which to find the local maxima
min	if this is set to 1, find local minima instead

Details

Uses C code in localmaxC.c

Value

matrix with 1 at the maxima and NA elsewhere

SlidingHistEqualC *INTERNAL Computes the sliding window histogram equalization of a matrix*

Description

INTERNAL Computes the sliding window histogram equalization of a matrix

Usage

```
SlidingHistEqualC(mat, radius, fullmax = 4096)
```

Arguments

mat	the matrix to equalize
radius	the radius of the sliding window (total window size is a square window with sides 2*radius+1)
fullmax	the maximum value in the equalized image

Details

Uses C code in slidinghistequalC.c

Value

The equalized matrix

SparseToMatrix *INTERNAL Converts a sparse mask to a matrix mask*

Description

INTERNAL Converts a sparse mask to a matrix mask

Usage

```
SparseToMatrix(sm, ny = 128, nx = 128, background = NA)
```

Arguments

sm	the sparse representation of the mask (a vector whose positive values are the indices of the mask pixels)
ny	the number of rows of the matrix mask
nx	the number of columbs of the matrix mask
background	the value to put in the non-mask pixels of the matrix

Value

A matrix of dimension (ny, nx) with 1's in the mask pixels and background elsewhere

TranslateFFT *Shifts an image by the given (fractional pixel) amounts*

Description

Shifts an image by the given (fractional pixel) amounts

Usage

```
TranslateFFT(img, xshift, yshift, fdomain = FALSE)
```

Arguments

img	the image (matrix) to shift
xshift	the amount to shift the in x dimension (columns)
yshift	the amount to shift in the y dimension (rows)
fdomain	is the image given in the Fourier domain? It will be returned in the same domain as given (passing in the Fourier domain is helpful to reduce superfluous transforms if performing additional operations in the Fourier domain).

Details

Uses the shift theorem to shift the given image by transforming to the Fourier domain. The shift can be sub-pixel, resulting in Fourier interpolation.

Value

the shifted image (matrix)

`ViewCI` *Opens the GUI viewer to manipulate the segmentation process.*

Description

Opens the GUI viewer to manipulate the segmentation process.

Usage

```
ViewCI(dbController = NULL)
```

Arguments

`dbController`	if specified, the viewer opens with the given dbController (looking in the directories stored in that object)

Value

NULL