Package ‘CRImage’

April 14, 2017

Type Package

Title CRImage a package to classify cells and calculate tumour cellularity

Version 1.22.0

Date 2012-10-01

Author
Henrik Failmezger <failmezger@mpipz.mpg.de>, Yinyin Yuan <Yinyin.Yuan@cancer.org.uk>, Oscar Rueda <oscar.rueda@cancer.org.uk>, Florian Markowetz <Florian.Markowetz@cancer.org.uk>

Maintainer Henrik Failmezger <failmezger@mpipz.mpg.de>, Yinyin Yuan

<Yinyin.Yuan@cancer.org.uk>

Description CRImage provides functionality to process and analyze images, in particular to classify cells in biological images. Furthermore, in the context of tumor images, it provides functionality to calculate tumour cellularity.

License Artistic-2.0

LazyLoad yes

Imports MASS, e1071, foreach, sgeostat

Depends EBImage, DNAcopy, aCGH

Collate plotCorrectedCN.R correctCopyNumber.R writeDensityImage.R convertRGBToHSV.R convertHSVToRGB.R imageCompression.R createBinaryImage.R colorCorrection.R searchStructures.R segmentStructures.R classifyStructures.R numberOfNeighbors.R segmentCytoplasma.R segmentImage.R createClassifier.R kernelSmoother.R paintCells.R classifyCells.R determineCellularity.R calculateCellularity.R findSlices.R parseFinalScan.R classificationAperio.R processAperio.R Phansalkar_threshold.R SauvolaThreshold.R calculateMeanStdTarget.R convertLABToRGB.R convertRGBToLAB.R localORThreshold.R oregonThreshold.R localThreshold.R calculateOtsu.R classifyPen.R getImageDistance.R hist3d.R labelCells.R plotImage.R

biocViews CellBiology, Classification

NeedsCompilation no

R topics documented:

`CRImage-package` *CRImage is a package to analyze images and classify cells.*

Description

CRImage allows classification of cells in biological images. It offers methods to segment cells or cell nuclei in biological images for example HE stained images. It offers methods to create a classifier and to classify cells in these images. Furthermore it allows the calculation of tumour cellularity for large microscope images.

CRImage makes use of the image processing package EBImage, which uses the 'ImageMagick' library for image I/O operations and the 'GTK' library to display images.

Details

Package:	CRImage
Type:	Package
Version:	1.0
Date:	2010-04-27
License:	LGPL Version 2 or later
LazyLoad:	yes

Package content

Image processing methods:

- calculateThreshold

- segmentImage

Classification:

- createTrainingSet
- createClassifier
- classifyCells

Tumour cellularity

- calculateCellularity
- processAperio

Author(s)

Henrik Failmezger, <failmezger@mpipz.mpg.de>

Yinyin Yuan, <Yinyin.Yuan@cancer.org.uk>

Oscar Rueda, <oscar.rueda@cancer.org.uk>

Florian Markowetz, <florian.markowetz@cancer.org.uk>

CRI Cambridge

Li Ka Shing Centre

Robinson Way

Cambridge, CB2 0RE, UK

Ludwigs-Maximilians University of Munich

Examples

```
example(segmentImage)
example(createClassifier)
example(classifyImage)
```

`calculateCellularity` *Calculation of tumour cellularity*

Description

The function calculates the tumour cellularity of an image by counting tumour and non tumour cells.

Usage

```
calculateCellularity(filename="",image=NA,classifier=NULL,cancerIdentifier=NA,KS=FALSE,maxShape=
```

Arguments

`filename`	A path to an image file.
`image`	If filename is undefined, an Image object
`classifier`	A SVM object, created with createClassifier or directly with the package e1071
`cancerIdentifier`	A string which describes, how the cancer class is named.
`KS`	Apply kernel smoother?
`maxShape`	Maximum size of cell nuclei
`minShape`	Minimum size of cell nuclei
`failureRegion`	minimum size of failure regions
`colors`	Colors to paint the classes
`threshold`	Which threshold should be uses, "otsu" or "phansalkar"
`classesToExclude`	Should a class be excluded from cellularity calculation?
`numWindows`	Number of windows for the threshold.
`classifyStructures`	Use hierarchical classification. If yes a pixel classifier has to be defined.
`pixelClassifier`	A SVM to classify pixel based on their color values. Needed if hierarchical classification should be applied.
`ksToExclude`	These classes are excluded from kernel smoothing.
`densityToExclude`	This class is excluded from cellularity calculation.
`numDensityWindows`	Number of windows for the density plot.

Details

The method calculates tumour cellularity of an image. The cells of the image are classified and the cellularity is: numTumourCells/numPixel. Furthermore the number of cells of the different classes are counted. A heatmap of cellularity is created. The image is divided in 16 subwindows and cellularity is calculated for every subwindow. Green in the heatmaps indicates strong cellularity, white low cellularity.

Value

A list containing

`cellularity values`	a vector, the n first values indicate the n numbers of cells in the n classes, the n + 1th value indicates the tumour cellularity, The n + 2th value is the ratio of tumour cells by all cells
`cancerHeatmap`	Heatmap of cancer density

Author(s)

Henrik Failmezger, failmezger@mpipz.mpg.de

Examples

```
t = system.file("extdata", "trainingData.txt", package="CRImage")
#read training data
trainingData=read.table(t,header=TRUE)
#create classifier
classifier=createClassifier(trainingData)[[1]]
#calculation of cellularity
f = system.file("extdata", "exImg.jpg", package="CRImage")
exImg=readImage(f)
cellularity=calculateCellularity(classifier=classifier,filename=f,KS=TRUE,maxShape=800,minShape=40,failureR
```

`calculateMeanStdTarget`
Calculates Mean and Standard deviation of an image

Description

Mean and SD calculation

Usage

```
calculateMeanStdTarget(imgT)
```

Arguments

`imgT`	the Image to calculate.

Details

Mean and SD

Value

Vector with mean and standard deviation.

Author(s)

Henrik Failmezger, failmezger@cip.ifi.lmu.de

Examples

```
#read the target image
f1= system.file("extdata", "exImg2.jpg", package="CRImage")
targetImage=readImage(f1)
#read the image whose color values should be adapted
f2= system.file("extdata", "exImg3.jpg", package="CRImage")
imgToConvert=readImage(f2)
#calculate mean and standard deviation of target color channels
mst=calculateMeanStdTarget(targetImage)
# create a white pixel mask
```

```
whitePixelMask=imgToConvert[,,1]>0.85 & imgToConvert[,,2]>0.85 & imgToConvert[,,3]>0.85
#adapt color channels of image
imgCorrected=colorCorrection(imgToConvert,mst,whitePixelMask)
```

`calculateOtsu` *Does Otsu thresholding*

Description

The function applies Otsu thresholding on the image.

Usage

```
calculateOtsu(allGreyValues)
```

Arguments

`allGreyValues`	Vector of grey values.

Details

The function calculates a value which separates the grey value histogram the best in foreground and background.

Value

the threshold

Author(s)

Henrik Failmezger, failmezger@cip.ifi.lmu.de

References

Nobuyuki Otsu: A threshold selection method from grey level histograms. In: IEEE Transactions on Systems, Man, and Cybernetics. New York 9.1979, S.62-66. ISSN 1083-4419

See Also

calculateThreshold localOtsuThreshold

Examples

```
f1= system.file("extdata", "exImg2.jpg", package="CRImage")
print(f1)
img=readImage(f1)
print(img)
#convert to grayscale
imgG=EBImage::channel(img,'grey')
#threshold value
t=calculateOtsu(as.vector(imgG))
```

`classifyCells` *A function to classify cells*

Description

The function classifies cells and paints the different class types in the image.

Usage

```
classifyCells(classifier,filename="",image=NA,segmentedImage=NA,featuresObjects=NA,paint=TRUE,KS
```

Arguments

`classifier`	A Support Vector Machine created by createClassifier or directly by the package e1071
`filename`	A path to an image file.
`image`	An 'Image' object or an array.
`segmentedImage`	An 'Image' object or an array.The corresponding segmented image (created by segmentImage)
`featuresObjects`	Cell feature file of the segmentedImage (created by segmentImage)
`paint`	If true, the classified cells are painted with different colors in the image
`KS`	Use Kernel Smoohter in classification?
`cancerIdentifier`	A string which describes, how the cancer class is named.
`maxShape`	Maximum size of cell nuclei
`minShape`	Minimum size of cell nuclei
`failureRegion`	minimum size of failure regions
`colors`	Colors to paint the classes
`classesToExclude`	Which class should be excluded?
`threshold`	Which thresholding method should be used, "otsu" or "phansalkar"
`numWindows`	Number of windows to use for thresholding.
`structures`	If the image is already segmented, structures can be inserted to enable hierarchical classification.
`classifyStructures`	Use hierarchical classification. If yes a pixel classifier has to be defined.
`pixelClassifier`	A SVM to classify pixel based on their color values. Needed if hierarchical classification should be applied.
`ksToExclude`	These classes are excluded from kernel smoothing.

Details

The kernels smoother improves the classification for cells which are likely to occur in clusters, like tumour cells. The kernel smoothing method can only be applied for two classes. If there are more classes only the normal svm without kernel smoothing is applied. Different classes are labeled with different colors in the image.

Value

A list with

comp1	classes
comp2	Classes, painted in the image, if paint was true

Author(s)

Henrik Failmezger, failmezger@mpipz.mpg.de

Examples

```
t = system.file("extdata", "trainingData.txt", package="CRImage")
#read training data
trainingData=read.table(t,header=TRUE)
#create classifier
classifier=createClassifier(trainingData)[[1]]
#classify cells
f = system.file("extdata", "exImg.jpg", package="CRImage")
classesValues=classifyCells(classifier,filename=f,KS=TRUE,maxShape=800,minShape=40,failureRegion=2000)
```

`colorCorrection` *Color transfer between images.*

Description

The colors of one image are adapted to the colors of a target image.

Usage

```
colorCorrection(imgO, meanStdTarget,whiteMask = c())
```

Arguments

imgO	The image who's colors should be adapted
meanStdTarget	Array with mean and standard deviation of the target image.
whiteMask	Boolean mask of white pixel in the image. These pixels are excluded from color correction.

Details

Mean and standard deviation of the target image can be calculated using the function calculate-MeanStdTarget.

Value

The image with adapted colors.

Author(s)

Henrik Failmezger, failmezger@cip.ifi.lmu.de

References

Reinhard, E.; Adhikhmin, M.; Gooch, B.; Shirley, P.; , "Color transfer between images," Computer Graphics and Applications, IEEE , vol.21, no.5, pp.34-41, Sep/Oct 2001 doi: 10.1109/38.946629 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=946629&isnumber=20481

See Also

calculateMeanStandardTarget

Examples

```
#read the target image
f1= system.file("extdata", "exImg2.jpg", package="CRImage")
targetImage=readImage(f1)
#read the image whose color values should be adapted
f2= system.file("extdata", "exImg3.jpg", package="CRImage")
imgToConvert=readImage(f2)
#calculate mean and standard deviation of target color channels
mst=calculateMeanStdTarget(targetImage)
# create a white pixel mask
whitePixelMask=imgToConvert[,,1]>0.85 & imgToConvert[,,2]>0.85 & imgToConvert[,,3]>0.85
#adapt color channels of image
imgCorrected=colorCorrection(imgToConvert,mst,whitePixelMask)
```

---

convertHSVToRGB *Conversion from HSV color space to RGB color space*

Description

The function converts images in the HSV colour space to the RGB colour space.

Usage

```
convertHSVToRGB(imgHSV)
```

Arguments

`imgHSV`	An 'Image' object or an array in the HSV colour space.

Details

Standard colour space conversion.

Value

An array in the RGB colour space.

Author(s)

Henrik Failmezger, failmezger@cip.ifi.lmu.de

See Also

convertRGBToHSV convertRGBToLAB convertLABToRGB

Examples

```
f= system.file("extdata", "exImg.jpg", package="CRImage")
img=readImage(f)
#conversion to RGB color space
imgRGB=convertHSVToRGB(img)
```

---

convertLABToRGB *Conversion of LAB colour space to RGB colour space*

Description

Color space conversion.

Usage

```
convertLABToRGB(imgLAB)
```

Arguments

`imgLAB`	LAB channel vectors.

Details

Color space conversion

Value

RGB channel vectors.

Author(s)

Henrik Failmezger, failmezger@cip.ifi.lmu.de

Examples

```
f= system.file("extdata", "exImg.jpg", package="CRImage")
img=readImage(f)
#conversion to HSV color space
imgRGB=convertLABToRGB(img)
```

`convertRGBToHSV` *Conversion from RGB color space to HSV color space*

Description

The RGB Image is converted to an HSV image.

Usage

```
convertRGBToHSV(img)
```

Arguments

`img`	The RGB image

Details

The entries of the array are Hue, Saturation and Value.

Value

The image in HSV color space.

Author(s)

Henrik Failmezger, failmezger@cip.ifi.lmu.de

See Also

convertHSVToRGB convertRGBToLAB convertLABToRGB

Examples

```
f= system.file("extdata", "exImg.jpg", package="CRImage")
img=readImage(f)
#conversion to HSV color space
imgHSV=convertRGBToHSV(img)
```

`convertRGBToLAB` *Converts RGB to LAB color space.*

Description

Conversion of Color spaces.

Usage

```
convertRGBToLAB(imgT)
```

Arguments

`imgT`	The RGB image.

Details

Color space conversion

Value

The image in LAB color space.

Author(s)

Henrik Failmezger, failmezger@cip.ifi.lmu.de

Examples

```
f= system.file("extdata", "exImg.jpg", package="CRImage")
img=readImage(f)
#conversion to LAB color space
imgLAB=convertRGBToLAB(img)
```

`correctCopyNumber` *Allelic Copy Number correction for cellularity*

Description

This function segments copy number and corrects log-ratios (LRR) and beta allele frequencies (BAF) values for cellularity.

Usage

```
correctCopyNumber(arr="Sample1", chr=NULL, p=NULL, z=NULL, min.value=-5)
```

Arguments

`arr`	Name of the array.
`chr`	Chromosome to run. If `NULL`, all chromosomes are run.
`p`	Percentage of tumoural cells.
`z`	Copy Number Data. Must be a dataframe with the following columns: Name (id of the probe), Chr (chromosome), Pos (position), LRR (log ratios) and BAF (beta allele frequencies).
`min.value`	Value assigned to the probes that have 0 copies after correction.

Details

The data.frame `z` must contain only SNP probes, that is probes with both LRR and BAF values. It is recommended that all replicated probes are merged so the positions are unique. This function calls `DNAcopy` to segment the LRR and then correct the segmented profiles for normal contamination according to the method described in the reference below (see for details).

Value

A list with 2 components:

`y`	a data.frame with as many rows as probes containing the following variables: Chrom (chromosome), Pos (position), Orig.LRR (LRR before correction) Orig.BAF (BAF before correction), Corr.LRR (LRR after cellularity correction) and Corr.BAF (BAF after correction)
`seg`	a data.frame with the segmented data. Contains the following columns: ID (name of the array), chrom (chromosome), loc.start (start of the region), loc.end (end of the region), num.mark (number of probes in the region), seg.mean (LRR of the region), BAF (BAF of the regions), num.BAF (number of SNP probes in the region), Sa (estimated absolute copy number for the first allele), Sb (estimated absolute copy number for the first allele), LRR.tum (corrected LRR for the region), BAF.tum (corrected BAF for the region).

Note

Includes an adaptation of `aCGH` `mergeLevels` function to fix a problem with `ansari.test`.

Author(s)

Oscar M. Rueda, rueda.om@gmail.com

References

Yuan, Y et al. Quantitative image analysis of cellular heterogeneity in primary breast tumors enriches genomic assays. In prep.

Examples

```
LRR <- c(rnorm(100, 0, 1), rnorm(10, -2, 1), rnorm(20, 3, 1),
       rnorm(100,0, 1))
BAF <- c(rnorm(100, 0.5, 0.1), rnorm(5, 0.2, 0.01), rnorm(5, 0.8, 0.01), rnorm(10, 0.25, 0.1), rnorm(10, 0.75
       rnorm(100,0.5, 0.1))

Pos <- sample(x=1:500, size=230, replace=TRUE)
Pos <- cumsum(Pos)
Chrom <- rep(1, length(LRR))
z <- data.frame(Name=1:length(LRR), Chrom=Chrom, Pos=Pos, LRR=LRR, BAF=BAF)
res <- correctCopyNumber(arr="Sample1", chr=1, p=0.75, z=z)
```

`createBinaryImage` *Thresholding*

Description

Creates a binary image from a grayscale image by thresholding.

Usage

```
createBinaryImage(imgG,img=NULL,method="otsu",threshold=NULL,numWindows=1,whitePixelMask=c())
```

Arguments

`img`	An Image object or an array.
`imgG`	The grey valued Image object.
`method`	Either "otsu" or "phansalkar"
`threshold`	Fixed threshold
`numWindows`	Number of windows to use for threshold calculation.
`whitePixelMask`	Boolean mask of white pixels, if they should be excluded from thresholding

Details

The functions returns the binary image resulting from the thresholding. If threshold is defined, all pixels smaller than this value will be fixed to 1 all other values will be set to 0. If threshold is undefined, the thresholding value is calculated automatically using 'otsu' or 'phansalkar' thresholding.

The function 'otsu' does Otsu thresholding on the grey level histograms of the image. The function 'phansalkar' does thresholding using the mean and standard deviation of a specified window. The thresholding is done on the RGB as well as the LAP color space and the results are ORed. The window size is dim(img)/numWindows. White pixel can be excluded from thresholding (e.g. if white is background) by defining a whitePixelMask

Value

The binary image.

Author(s)

Henrik Failmezger, failmezger@lmb.uni-muenchen.de

References

Neerad Phansalkar, Sumit More, Ashish Sabale, Dr. Madhuri Joshi, "Adaptive Local Thresholding for Detection of Nuclei in Diversly Stained Cytology Images," 2011 IEEE International Conference in Communications and Signal Processing (ICCSP), pp. 218, 10 Feb. 2011

Nobuyuki Otsu: A threshold selection method from grey level histograms. In: IEEE Transactions on Systems, Man, and Cybernetics. New York 9.1979, S.62-66. ISSN 1083-4419

Examples

```
f= system.file("extdata", "exImg.jpg", package="CRImage")
img=readImage(f)
#conversion to grayscale
imgG=EBImage::channel(img,"gray")
imgB=createBinaryImage(imgG,img=img,method="otsu",numWindows=4)
#white pixel mask
whitePixelMask=img[,,1]>0.85 & img[,,2]>0.85 & img[,,3]>0.85
#exclude white pixels from thresholding
imgB=createBinaryImage(imgG,img=img,method="otsu",numWindows=4,whitePixelMask)
#phansalkar threshold
imgB=createBinaryImage(imgG,img=img,method="phansalkar",numWindows=4)
```

createClassifier *Construction of a classifier*

Description

Creates a classifier for a training set.

Usage

```
createClassifier(trainingData, cross = FALSE)
```

Arguments

trainingData	A table, created by segmentImage with manually added classes.
cross	Does 10-fold cross validation to test the classifiers performance.

Details

Topological features include the density of cells and the size of the surrounding cytoplasma of a cell. These features depend on the size of the image. If training image and the image to classify have different size, these features can fool the classification and should not be enabled.

Value

A List containing:

classifier	The classifier
performance	cross validation performance

Author(s)

Henrik Failmezger, failmezger@mpipz.mpg.de

See Also

'createTrainingSet','classifyCells'

Examples

```
f = system.file("extdata", "trainingData.txt", package="CRImage")
#read training data
trainingData=read.table(f,header=TRUE)
#create classifier
classifier=createClassifier(trainingData)[[1]]
```

labelCells *Interactive Session for cell labeling*

Description

The functions creates an interactive session in order to label cells with their classes. The labeled cells can be used as training set for the classifier. Note!! This is until now only tested for MacOsX.

Usage

```
labelCells(img, segmentedImage, classes, classColours, nblocks = 3, labeledPoints = NULL, filename
```

Arguments

`img`	The image.
`segmentedImage`	The segmented image.
`classes`	The possible class labels.
`classColours`	The colors for the class labels.
`nblocks`	The image can be separated in several blocks, as zooming is not possible.
`labeledPoints`	Labeled cells from a previous training session.
`filename`	The table of labeled cells is saved at this location.
`filenameImage`	The image with the labeled cells is saved at this location.
`transformCoordinates`	deprecated

Details

Use the keys: a: In order to add a label to a cell. d: In order to delete a label from a cell. c: To switch between classes. q: To quit the interactive session. r: To refresh the session (labeled cells will be shown after refreshing)

Value

A table with columns: index: the index of the cell in the segmented image. x: x-coordinate of the cell y: y-coordinate of the cell classCell: Label of the cell xLocal: Local x coordinate in the subimage(block) yLocal: Local y coordinate in the subimage(block) block: Block number in which the cell arises.

Author(s)

Henrik Failmezger

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
```

`plotCorrectedCN` *Plot CN profiles corrected for cellularity*

Description

This function takes the result of a call to `correctCopyNumber` and plots the results.

Usage

```
plotCorrectedCN(CN, chr=NULL)
```

Arguments

`CN`	object result of a call to `correctCopyNumber`.
`chr`	chromosome to plot.

Details

A panel with four plots is created. The top panel shows LRR (with DNAcopy segmentation overlayed) and BAF before correction and the bottom panel shows the plots after correction.

Value

No value is returned.

Author(s)

Oscar M. Rueda, rueda.om@gmail.com

References

Yuan, Y et al. Quantitative image analysis of cellular heterogeneity in primary breast tumors enriches genomic assays. In prep.

Examples

```
LRR <- c(rnorm(100, 0, 1), rnorm(10, -2, 1), rnorm(20, 3, 1),
       rnorm(100,0, 1))
BAF <- c(rnorm(100, 0.5, 0.1), rnorm(5, 0.2, 0.01), rnorm(5, 0.8, 0.01), rnorm(10, 0.25, 0.1), rnorm(10, 0.75
       rnorm(100,0.5, 0.1))

Pos <- sample(x=1:500, size=230, replace=TRUE)
Pos <- cumsum(Pos)
Chrom <- rep(1, length(LRR))
z <- data.frame(Name=1:length(LRR), Chrom=Chrom, Pos=Pos, LRR=LRR, BAF=BAF)
res <- correctCopyNumber(arr="Sample1", chr=1, p=0.75, z=z)
plotCorrectedCN(res, chr=1)
```

`processAperio` *Cellularity Calculation of Aperio TX Scanner*

Description

Procession of Aperio TX Slides.

Usage

```
processAperio(classifier=classifier,inputFolder=inputFolder,outputFolder=outputFolder,identifier
```

Arguments

`classifier`	The classifier.
`inputFolder`	The path to the image folder.
`outputFolder`	The path to the output folder.
`identifier`	The identifier of the files ("Ss" or "Da")
`numSlides`	The number of sections in the image.
`cancerIdentifier`	The identifier of the cancer class
`classOther`	deprecated
`maxShape`	Maximum size of cell nuclei
`minShape`	Minimum size of cell nuclei
`failureRegion`	minimum size of failure regions
`slideToProcess`	Set this parameter if only a certain slide should be processed
`KS`	Apply Kernel Smoother?
`colors`	Colors to paint the classes
`classesToExclude`	Which class should be excluded?
`threshold`	Which thresholding method should be used, "otsu" or "phansalkar" possible
`numWindows`	Number of windows to use for thresholding.
`colorCorrection`	deprecated
`classifyStructures`	Use hierarchical classification. If yes a pixel classifier has to be defined.
`ksToExclude`	These classes are excluded from kernel smoothing.
`pixelClassifier`	A SVM to classify pixel based on their color values. Needed if hierarchical classification should be applied.
`densityToExclude`	This class is excluded from cellularity calculation.
`numDensityWindows`	Number of windows for the density plot.

`resizeFactor`	Specifies the size of the cell density image. If this variable is not defined, the size of the thumbnail is used for the cell density image, else the size is calculated by size(thumbnail)*resizeFactor. The thumbnail is the small overview image, created by the Aperio software.
`plotCellTypeDensity`	Plot the density of different cell types?
`greyscaleImage`	Color channel of the RGB image that should be used for thresholding
`penClassifier`	Classifier to exclude low quality images (will be part of next release)
`referenceHist`	Colour Histogram of a reference image that can be used to calculate the quality of the recent image. (will be part of next release)
`fontSize`	will be part of next release

Details

The function processes images of Aperio TX scanners. The images have to be saved in the CWS format.

Value

Four folders are created in the output folder.

`Files`	Cellularity values and cell numbers are saved in the file
`classifiedImage`	Subimages with labeled tumour and non tumour cells
`tumourDensity`	Cancer heatmaps for every subimage
`cellCoordinates`	Coordinates and cell class for every cell in the subimage
`resizeFactor`	Size of the cellularity density image, calculated by size(thumbnail) * resizeFactor. Whereas the thumbnail is the small overview image produced by Aperio.

Author(s)

Henrik Failmezger, failmezger@mpipz.mpg.de

Examples

```
#t = system.file("extdata", "trainingData.txt", package="CRImage")
#read training data
#trainingData=read.table(t,header=TRUE)
#create classifier
#classifier=createClassifier(trainingData,topo=FALSE)[[1]]
#classify aperio
#f = system.file("extdata",  package="CRImage")
#f=file.path(f,"8905")
#dir.create("AperiOutput")
#takes long time!

f = system.file("extdata",  package="CRImage")
fc=file.path(f,"testClassifier")
load(fc)
fp=file.path(f,"pixelClassifier")
load(fp)
pixelClassifier=model
```

```
pathToImage=file.path(f,"8905")

pathToOutput="" #specify an output folder here

#processAperio(classifier=classifier,inputFolder=pathToImage,outputFolder=pathToOutput,identifier="Da",numS
```

`SauvolaThreshold` *Do Sauvola thresholding*

Description

Thresholding method using mean and standard deviation.

Usage

```
SauvolaThreshold(allGreyValues)
```

Arguments

`allGreyValues`	Vector of gray values.

Details

A threshold for the gray values is returned

Value

The threshold.

Author(s)

Henrik Failmezger, failmezger@cip.ifi.lmu.de

References

J. Sauvola, M. Pietikainen, "Adaptive Document Image Binarization," Pattern Recognition, vol. 33, 225-236, 2000

See Also

createBinaryImage

Examples

```
f1= system.file("extdata", "exImg2.jpg", package="CRImage")
print(f1)
img=readImage(f1)
print(img)
#convert to grayscale
imgG=EBImage::channel(img,'grey')
#threshold value
t=SauvolaThreshold(as.vector(imgG))
```

segmentImage *Segmentation of an image*

Description

The function segments cells or cell nuclei in the image.

Usage

```
segmentImage(filename="",image=NA,maxShape=NA,minShape=NA,failureRegion=NA,threshold="otsu",numW
```

Arguments

`filename`	A path to an image
`image`	An 'image' object, if no filename is specified.
`maxShape`	Maximum size of cell nuclei
`minShape`	Minimum size of cell nuclei
`failureRegion`	minimum size of failure regions
`threshold`	Thresholding method, "otsu" or "phansalkar"
`numWindows`	Number of windows to use for thresholding.
`colorCorrection`	deprecated
`classifyStructures`	Segment structures in the image, if yes a pixel classifier has to be defined
`pixelClassifier`	A SVM which classifies RGB color values in foreground and background.
`greyscaleImage`	Channel of the RGB image, to use for thresholding, if 0 use a joined greyscale image.
`penClassifier`	Classifier to exclude low quality images(will be part of next release)
`referenceHist`	Color histogram of a reference image, that can be used to estimate the quality of the recent image (will be part of next release)

Details

The image is converted to greyscale and thresholded. Clutter is deleted using morphological operations. Clustered objects are separated using watershed algorithm. Segmented Cell nuclei, which exceed the maximum size are thresholded and segmented again. Cell nuclei which fall below the minimum size are deleted. Dark regions which exceed the parameter failureRegion are considered as artefacts and deleted. If the parameters are not defined, the operations will not be executed. Features are generated for every segmented object.

Value

A list is returned containing

`image`	The original image
`segmented image`	The segmented image

Author(s)

Henrik Failmezger, failmezger@cip.ifi.lmu.de

References

EBImage, 'http://www.bioconductor.org/packages/release/bioc/html/EBImage.html'

Examples

```
#segment image
#f = system.file('extdata' ,'exImg.jpg',package='CRImage')
#segmentationValues=segmentImage(f,maxShape=800,minShape=40,failureRegion=2000,threshold="otsu",numWindows=
#image=segmentationValues[[1]]
#segmentedImage=segmentationValues[[2]]
#imageFeatures=segmentationValues[[3]]
```

Index