# CRImage

October 25, 2011

---

`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@cip.ifi.lmu.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, cancerIdentifier, KS
```

## 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 |

### 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@cip.ifi.lmu.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]]
#calculation of cellularity
f = system.file("extdata", "exImg.jpg", package="CRImage")
exImg=readImage(f)
cellularityValues=calculateCellularity(f,classifier=classifier,cancerIdentifier="1",maxSh
```

---

## `calculateThreshold` *Thresholding*

---

### Description

Calculates the grey value which separates the grey-level image best in foreground and background. Tthe Otsu Method is used for calculating the threshold.

### Usage

```
calculateThreshold(allGreyValues)
```

### Arguments

`allGreyValues`
: a vector of grey values.

## Details

The optimal threshold is searched by a histogram separation method.

## Value

The calculated threshold.

## Author(s)

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

## References

Otsu, N. A threshold selection method from gray level histograms IEEE Trans. Systems, Man and Cybernetics, 1979, 9, 62-66

## Examples

```
f= system.file("extdata", "exImg.jpg", package="CRImage")
img=readImage(f)
#convert to grayscale
imgB=channel(img,"gray")
#find white pixels and exclude them from thresholding(if white is background)
indexWhitePixel=which(img[,,1]>0.85 &img[,,2]>0.85 & img[,,3]>0.85)
#calculate threshold
t=calculateThreshold(as.vector(imgB[-indexWhitePixel]))
#create binary image
imgB[imgB>t]=-1
imgB[imgB != -1]=0
imgB[imgB == -1]=1
```

---

# classificationAperio

*Classification of ScanScopeTX Slides.*

---

## Description

The slides are segmented and classified.

## Usage

```
classificationAperio(fileLocation, filename, pathToOutputFolderImgDir, classifie
```

## Arguments

| | |
|---|---|
| `fileLocation` | location of the image file |
| `filename` | name of the image file |
| `pathToOutputFolderImgDir` | path where the classified images are saved |
| `classifier` | the classifier object |
| `pathToOutputFolderImgDirFiles` | the path where the files with the cellularity values should be saved. |

`pathToOutputFolderImgDirImages`
: the path where the classified images are saved.

`pathToOutputFolderImgDirCellDensity`
: the path where the cancer heatmaps are saved

`blockSlice`
: which slide does the image have

`sliceColors`
: colors to label the classes

`sizeO`
: size of the image

`index`
: index of the image

`cancerIdentifier`
: label of the cancer identifier

`maxShape`
: Maximum size of cell nuclei

`minShape`
: Minimum size of cell nuclei

`failureRegion`
: minimum size of failure regions

`KS`
: Apply KernelSmoother?

## Details

The function is an internal function which is used by processAperio.

## Author(s)

Henrik Failmezger, failmezger@cip.ifi.lmu.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!
#processAperio(classifier=classifier,inputFolder=f,outputFolder="AperiOutput",identifier=
```

---

# 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, featur
```

### 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 |

### 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@cip.ifi.lmu.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 cells
f = system.file("extdata", "exImg.jpg", package="CRImage")
classesValues=classifyCells(classifier,filename=f,KS=TRUE,,maxShape=800,minShape=40,failu
```

`createClassifier` *Construction of a classifier*

## Description

Creates a classifier for a training set.

## Usage

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

## Arguments

| | |
|---|---|
| `trainingData` | A table, created by segmentImage with manually added classes. |
| `cross` | Does 10-fold cross validation to test the classifiers performance. |
| `topo` | Use topological features. |

## 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@cip.ifi.lmu.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,topo=FALSE)[[1]]
```

`createTrainingSet` *Construction of a training set*

## Description

Creates a training set for cell classification.

## Usage

```
createTrainingSet(filename = "", image = NA,maxShape = NA,minShape = NA,failureR
```

## Arguments

| | |
|---|---|
| `filename` | Path to an image file. |
| `image` | An 'Image' object, if filename is not specified. |
| `maxShape` | Maximum size of cell nuclei |
| `minShape` | Minimum size of cell nuclei |
| `failureRegion` | minimum size of failure regions |

## Details

The image is segmented. An image is created, in which every cell is labeled with a numberl. Furthermore, a table including the features of the cells is created. In order to create the training set, the table with the cell features has to be opened for instance in a spreadsheet program. Class values for the cells have to be inserted in the column 'class'. The corresponding cell in the image can be identified by the column 'index' (numbers in column index correspond to numbers in the image). Class values for different classes can be numbers or strings. Be careful, this function does not work on MacOsX because of font incompabilities.

## Value

A List containing:

| | |
|---|---|
| `labeledImage` | Image with labeled cells |
| `cellFeatures` | Table of the cell features. |

## Author(s)

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

## See Also

'createClassifier'

## Examples

```
f = system.file("extdata", "exImg.jpg", package="CRImage")
trainingValues=createTrainingSet(filename=f,maxShape=800,minShape=40,failureRegion=2000)
#display(trainingValues[[1]])
#trainingValues[[2]]
```

`determineCellularity` *Determination of cellularity*

## Description

The function is an internal function which calculates cellularity.

## Usage

```
determineCellularity(classes, classifiedCells, dimImg, img, imgW, indexWhitePixe
```

## Arguments

| | |
|---|---|
| `classes` | the classes |
| `classifiedCells` | the classified cells |
| `dimImg` | dimension of the image |
| `img` | the image |
| `imgW` | the segmented image |
| `indexWhitePixel` | the index of the white pixels |
| `cancerIdentifier` | the label of the cancer identifier |
| `classValues` | the class values |

## Details

The function is an internal function.

## Value

The calculated cellularity

## Author(s)

Henrik Failmezger, failmezger@cip.ifi.lmu.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]]
#calculation of cellularity
f = system.file("extdata", "exImg.jpg", package="CRImage")
exImg=readImage(f)
cellularityValues=calculateCellularity(f,classifier=classifier,cancerIdentifier="1",maxSh
```

`findSlices` *calculates the coordinates of the subimages*

## Description

internal function

## Usage

```
findSlices(imgFolder, pathToOutputFolder, numSlides)
```

## Arguments

| | |
|---|---|
| `imgFolder` | folder to the subimages |
| `pathToOutputFolder` | path to the output folder |
| `numSlides` | number of sections in the image |

## Details

internal function

## Value

the sections for every image

## Author(s)

Henrik Failmezger, failmezger@cip.ifi.lmu.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!
#processAperio(classifier=classifier,inputFolder=f,outputFolder="AperiOutput",identifier=
```

`kernelSmoother` *A kernel smoother to classify images*

## Description

internal function

## Usage

```
kernelSmoother(predictedClasses, cellCoordinates, segmentedImage, cancerIdentifi
```

## Arguments

| | |
|---|---|
| `predictedClasses` | the predicted classes |
| `cellCoordinates` | the coordinates of the cells |
| `segmentedImage` | the segmented image |
| `cancerIdentifier` | the label of the cancer identifier |
| `indexCells` | the index of the cells |

## Details

internal function

## Value

the classes calculated by the kernel smoother

## Author(s)

Henrik Failmezger, failmezger@cip.ifi.lmu.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 cells
f = system.file("extdata", "exImg.jpg", package="CRImage")
classesValues=classifyCells(classifier,filename=f,KS=TRUE,,maxShape=800,minShape=40,failu
```

## localThreshold *Internal function to do local thresholding.*

### Description

The function calculates a local threshold of an image using calculateThreshold.

### Usage

```
localThreshold(imgG, img)
```

### Arguments

| | |
|---|---|
| imgG | Greyscale image. |
| img | Image with colors. |

### Details

The function is an internal function.

### Value

The thresholded image.

### Author(s)

Henrik Failmezger, <email: failmezger@cip.ifi.lmu.de>

### Examples

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

## numberOfNeighbors *Internal function to calculate the number of neighbors of a cell.*

### Description

Calculates the number of neighbors of a cell nuclei in a distance of 50 pixel.

### Usage

```
numberOfNeighbors(img, cellCoordinates, allFeatures)
```

### Arguments

| | |
|---|---|
| `img` | The image. |
| `cellCoordinates` | Coordinates of the cells. |
| `allFeatures` | Features of the cells. |

### Details

The function is an internal function.

### Value

The number of neighbors for every cell.

### Author(s)

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

### Examples

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

---

## `paintCells` *Coloring of classified cells*

---

### Description

internal function

### Usage

```
paintCells(imgWT, img, classes, index, classValues)
```

### Arguments

| | |
|---|---|
| `imgWT` | segmented image |
| `img` | the image |
| `classes` | the classes |
| `index` | index |
| `classValues` | the class values |

### Details

internal function

### Author(s)

Henrik Failmezger,failmezger@cip.ifi.lmu.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 cells
f = system.file("extdata", "exImg.jpg", package="CRImage")
classesValues=classifyCells(classifier,filename=f,KS=TRUE,,maxShape=800,minShape=40,failu
```

---

## parseFinalScan *Internal function to parse CWS index files.*

---

### Description

The function parses the FinalScan.ini file, to a R readable format.

### Usage

```
parseFinalScan(file)
```

### Arguments

| | |
|---|---|
| file | CWS file |

### Details

The FinalScan.ini files keep information about the information of subimages in the whole image.

### Value

List with subimage names and corresponding slides.

### Author(s)

Henrik Failmezger, <email: failmezger@cip.ifi.lmu.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!
#processAperio(classifier=classifier,inputFolder=f,outputFolder="AperiOutput",identifier=
```

`processAperio` *Cellularity Calculation of Aperio TX Scanner*

## Description

Procession of Aperio TX Slides.

## Usage

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

## 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 |
| `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? |

## 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 |

## Author(s)

Henrik Failmezger, failmezger@cip.ifi.lmu.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!
#processAperio(classifier=classifier,inputFolder=f,outputFolder="AperiOutput",identifier=
```

---

## segmentCytoplasma *Internal function to segment cell cytoplasma*

---

### Description

The functions segments the cytoplasm of a cell.

### Usage

```
segmentCytoplasma(img, imgW, indexWhitePixel, imgG, index, hF)
```

### Arguments

| | |
|---|---|
| img | The image. |
| imgW | The segmented image. |
| indexWhitePixel | Index of white pixels. |
| imgG | The greyscale image. |
| index | THe index of the cells. |
| hF | The hull features of the cells |

### Details

The function is an internal function.

### Value

The segmented cytoplasma

### Author(s)

Henrik Failmezger, <email: failmezger@cip.ifi.lmu.de>

### Examples

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

---

## 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, failureRegi
```

### 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 |

### 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)
image=segmentationValues[[1]]
segmentedImage=segmentationValues[[2]]
imageFeatures=segmentationValues[[3]]
```

# Index