CRImage, a package for classifying cells and caculating tumour cellularity

Henrik Failmezger

April 21, 2010

1 Load the package

The package is loaded by the following command:

```
> library(CRImage)
```

See the functions of the package EBImage to read, write and manipulate an image.

2 Threshold an image

To create a binary image of a coloured image the function calculateThreshold can be used. This function calculates Otsu threshold on an image.

```
> f = system.file("data", "exImg2.jpg", package = "CRImage")
> exImgB = readImage(f)
> indexWhitePixel = which(exImgB[, , 1] > 0.85 & exImgB[, , 2] >
+     0.85 & exImgB[, , 3] > 0.85)
> exImgB = channel(exImgB, "gray")
> t = calculateThreshold(as.vector(exImgB[-indexWhitePixel]))
> exImgB[which(exImgB >= t)] = 1
> exImgB[which(exImgB < t)] = 0
```

The image is read with `readImage`. `indexWhitePixel` finds white pixels in the image. These pixels are considered as background and excluded from thresholding. `channel(exImgB,"gray")` the image is converted to grayscale and thresholded with `calculateThreshold`. The binary image is created by finding pixels with gray values larger or smaller than the threshold and colouring them black or white.

3 Segment an image

An image can be segmented to find cells in the image, using the function `segmentImage`.

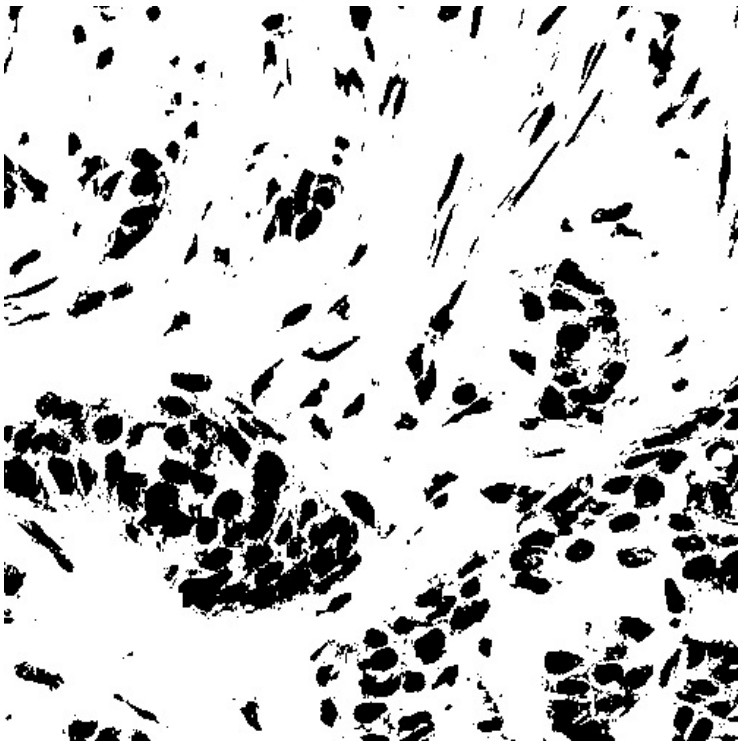

Figure 1: Thresholded image.

```
> f = system.file("data", "exImg2.jpg", package = "CRImage")
> segmentationValues = segmentImage(f)
```

The image is converted to grayscale and thresholded. Morphological opening is used to delete clutter and to smooth the shapes of the cells. THe watershed algorithm is used to separate clustered cells. `segmentationValues[[1]]` holds the original image, `segmentationValues[[2]]` holds the segmented image, `segmentationValues[[3]]` holds features, which were calculated for the segmented objects. The segmented objects can be drawn by the function `display(paintObjects(segmentedIma` (see EBImage).

4 Creating a training set

To classify cells in images first an approbiate training set has to be created.

```
f = system.file("data", "exImg.jpg", package="CRImage")
trainingValues=createTrainingSet(filename=f)
```

The list trainingValues returns two values. The first value is an image which can be displayed by:

```
 display(trainingValues[[1]])
```

is an image in which every segmented cell is numbered. The second value is an table with features for every cell:

```
index class      g.x       g.y g.s g.p     g.pdm    g.pdsd    g.effr...
1   <NA>148.2203  4.855932 118  36  5.721049 1.0130550  6.128668...
2   <NA>  160.2719  4.763158 114  38  5.659780 1.0331399  6.023896...
3   <NA>  183.7975  3.101266  79  35  4.593585 0.9575905  5.014627...
4   <NA>  196.3500  4.242857 140  43  6.424591 1.4433233  6.675581...
5   <NA>  271.5694  2.680556  72  29  4.504704 1.2490794  4.787307...
6   <NA>  338.5221  6.530973 113  35  5.601531 1.0849651  5.997418...
7   <NA>  456.0179  2.946429 112  39  5.726556 1.8101368  5.970821...
8   <NA>  556.3778  9.018519 270  81  9.196894 2.4687870  9.270581...
9   <NA>  575.1777 10.935950 484 101 11.959995 2.1378649 12.412171...
10   <NA>  592.7391  1.724638  69  41  5.145094 2.2782539  4.686511...
```

To create the training set class values for the cells have to be inserted in the column class:

```
index class      g.x       g.y g.s g.p     g.pdm    g.pdsd    g.effr
1   normal 148.2203  4.855932 118  36  5.721049 1.0130550  6.128668
2   malignant  160.2719  4.763158 114  38  5.659780 1.0331399  6.023896
```

The values in the column index match the numbers for the cells in the image. You can save the table as tab seperated by:

```
write.table(trainingData,file="path",sep="\t",rownames=F)
```

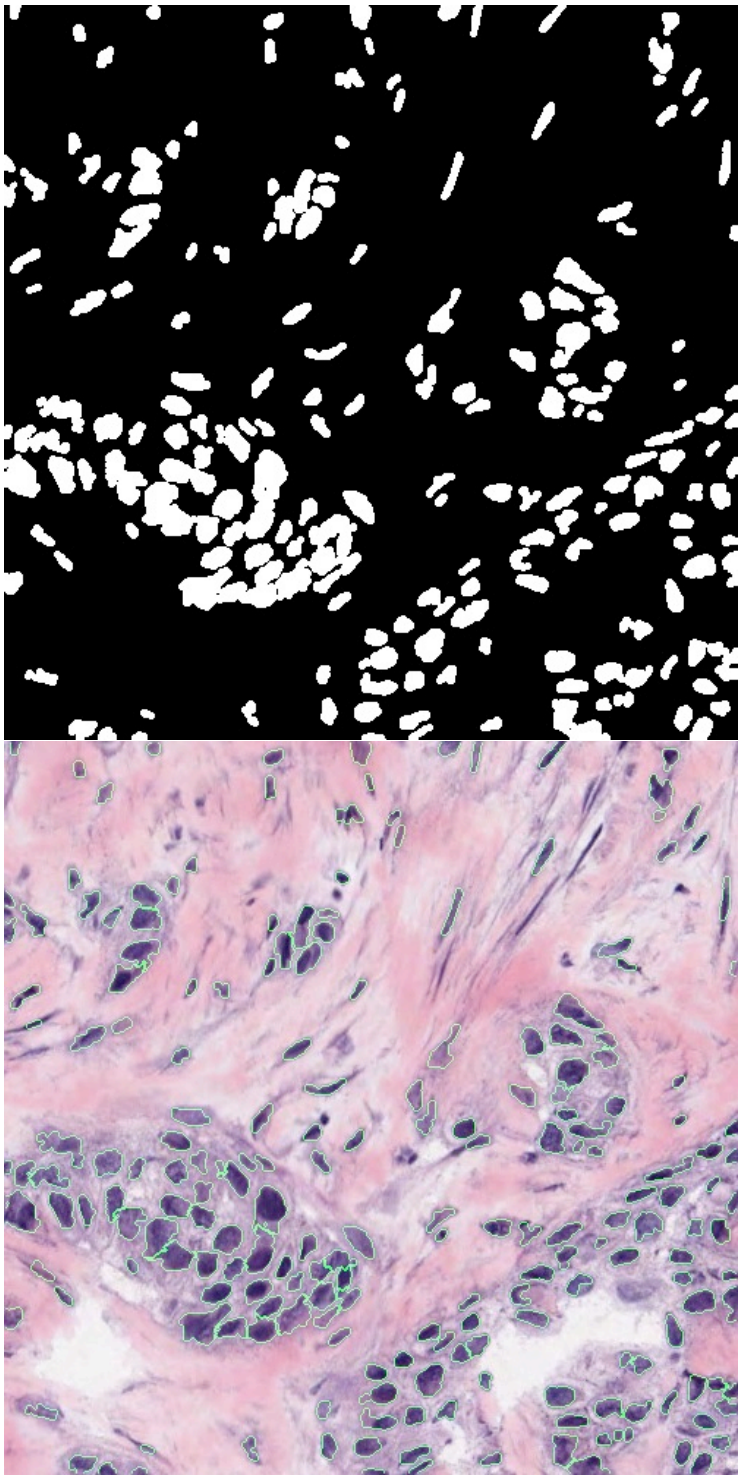

Figure 2: SegmentedImage.

and open it for example in a spreadsheet program, however be careful not to shift the columns. You do not have to assign a class value to every cell but do ensure that there are enough examples for every class. Class values can be numbers (1,2,3) or strings (normal, malignant). You can choose at most 10 classes. If you want to use the kernel smoothing approach for classification you can only choose two classes.

5 Creating the classifier

The command:

```
> f = system.file("data", "trainingData.txt", package = "CRImage")
> trainingData = read.table(f, header = TRUE)
> classifierValues = createClassifier(trainingData, topo = FALSE)
> classifier = classifierValues[[1]]
```

creates the classifier. The classifier is a Support Vector Machine provided by the package e1017.

6 Classification of cells

After having created the classifier, cells in another image can be classified.

```
> f = system.file("data", "exImg2.jpg", package = "CRImage")
> classValues = classifyCells(classifier, filename = f, KS = TRUE)
```

7 Calculation of cellularity

If tumour images are processed, the cellularity of the tumour can be calculated. The image is first segmented and the cell types are classified. Afterwards, the cellularity of the tumour is determined. The function needs to know, which class value should be the tumour class.

```
> t = system.file("data", "trainingData.txt", package = "CRImage")
> trainingData = read.table(t, header = TRUE)
> classifier = createClassifier(trainingData, topo = FALSE)[[1]]
> f = system.file("data", "exImg2.jpg", package = "CRImage")
> exImg = readImage(f)
> cellularityValues = calculateCellularity(f, classifier = classifier,
+     cancerIdentifier = "1")
```

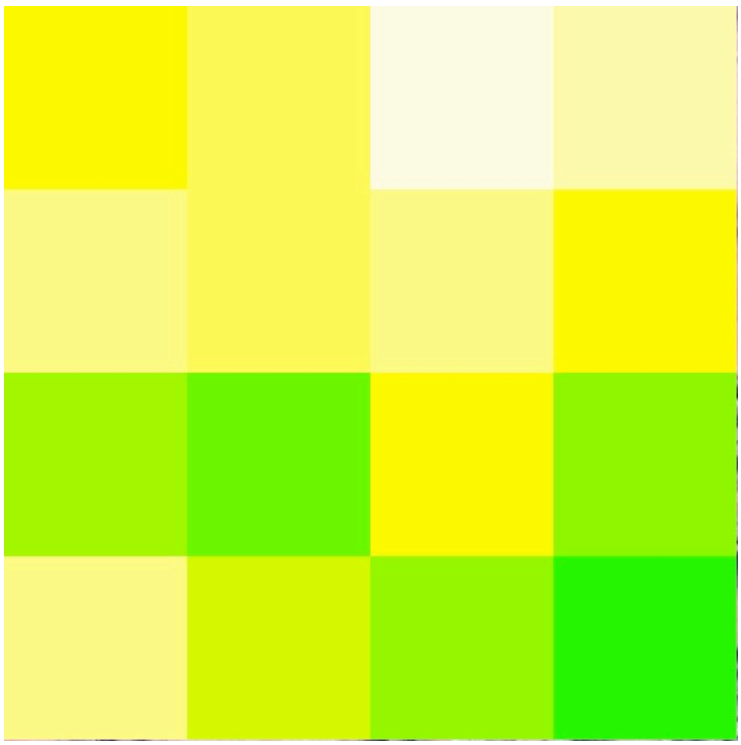

Figure 3: Heatmap of cellularity values.

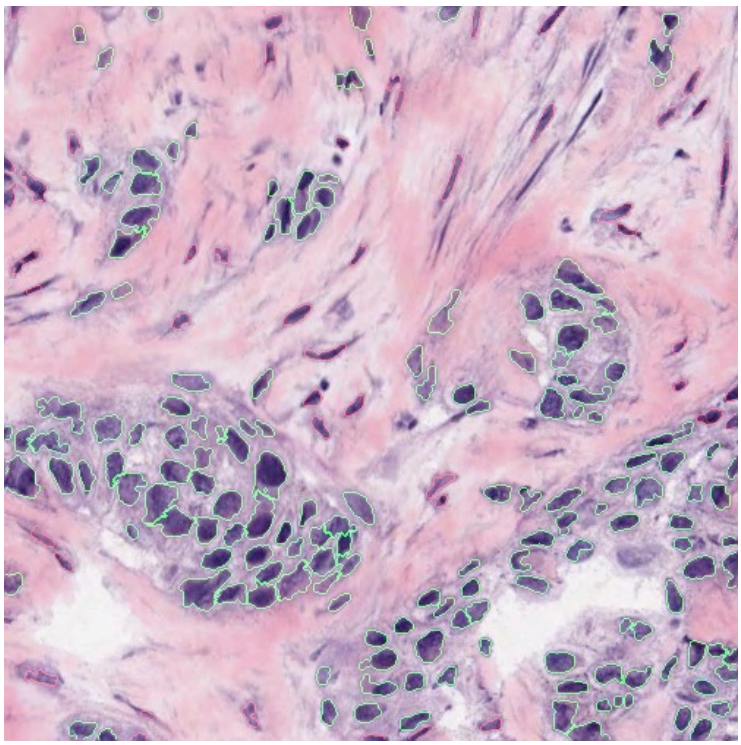

Figure 4: Classified Image.