FISHalyseR
Automated fluorescence in situ hybridisation quantification in R

Andreas Heindl, Karesh Arunakirinathan

E-mail: andreas.heindl@icr.ac.uk, akaresh88@gmail.com

Contents

1 Load the package

To load the package use the following command:

```
library(FISHalyseR)

## Loading required package:  EBImage
## Loading required package:  abind
##
## Attaching package:  'abind'
## The following object is masked from 'package:EBImage':
##
##    abind
## Warning:  replacing previous import 'EBImage::abind' by 'abind::abind' when loading 'FISHalyseR'
```

FISHalyseR utilises the functions of the R package EBImage to read, write and manipulate images.

2 Available image thresholding methods

2.1 Max Entropy Threshold

The Maximum Entropy algorithm is a thresholding technique similar to Otsu. It allows for binarising an image by maximising the inter-class entropy. The method enables reliable probe detection. It requires only a single input parameter which is the grayscale image. Figure 1 illustrates the Max Entropy Thresholding of an image containing FISH probes and nuclei.

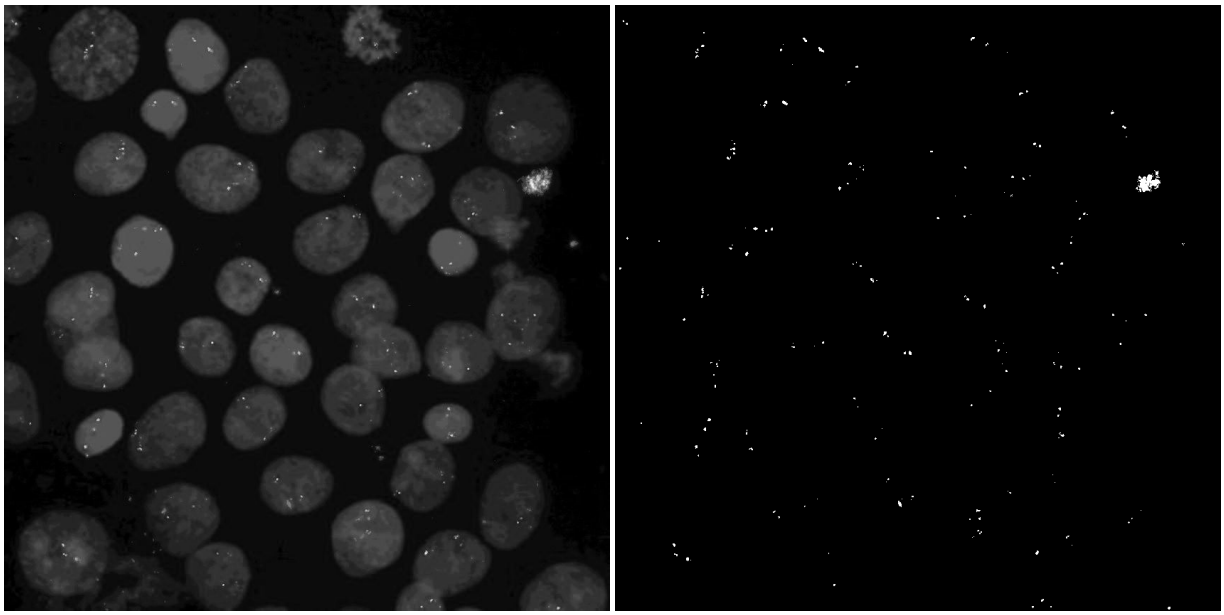

Figure 1. The left figure shows the original grayscale input image containing the nuclei and the FISH probes. Applying maximum entropy thresholding results in the binary image on the right side. The location of the FISH probes is indicated by white pixels.

```
f = system.file( "extdata", "SampleFISHgray.jpg", package="FISHalyseR")
img = readImage(f)

t = calculateMaxEntropy(img)
img[img<t] <- 0
img[img>=t] <- 1
```

2.2 Otsu threshold

Otsu's method (Figure 2) is a well studied thresholding techniques that calculates the optimal threshold separating two classes such that their intra-class variance is minimal. The method requires similar to the maximum entropy thresholding only a grayscale image as input parameter.

```
f = system.file( "extdata", "SampleFISHgray.jpg", package="FISHalyseR")
img = readImage(f)

t = calculateThreshold(img)
img[img<t] <- 0
img[img>=t] <- 1
```

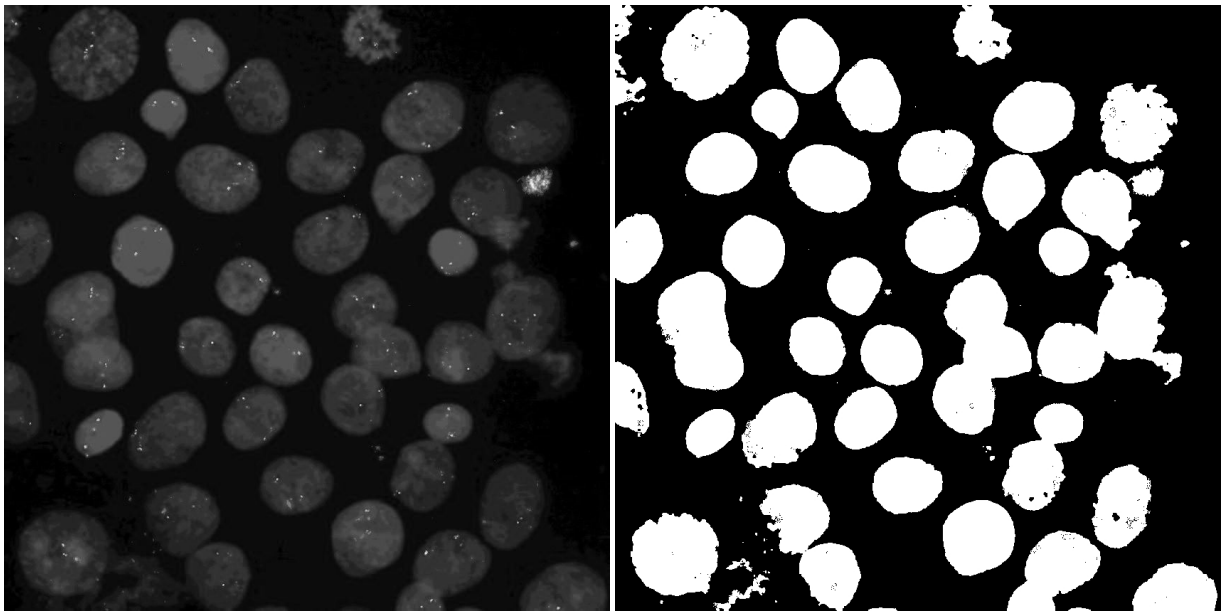

Figure 2. The image on the left side shows the original input image image containing nuclei. After thresholding it using Otsu's method a binary image is created indicating the location of each nucleus in white. Note that clutter and artefacts will we removed using the analyseParticles function during processing.

3 Analyse Particles

`analyseParticles` is used to clean up clutter and artefacts from binary images. The user can specify minimal and maximal area allowed for each connected component (e.g. nucleus, probe, etc.....). The example beneath illustrates how to remove components with an area smaller than 5 pixels and an area larger than 20 pixels. Figure 3 depicts the process with an example image containing FISH probes.

```
fProbes1 = paste(TempDir,'/',"exImgMaxEntropyProbes1.jpg",sep='')
img = readImage(fProbes1)

anaImg <- analyseParticles(img, 20, 5,0)
```

The function is also used to clean up the cell mask image. The example beneath shows a call of analyseParticles that removes all connected component with an area smaller than 1000 pixel and greater than 20000 pixels. The process is illustrated in Figure 4 using our cell mask from Figure 2.

```
f = paste(TempDir,'/',"exImgOtsuCellMask.jpg", sep='')
img = readImage(f)

anaImg <- analyseParticles(img, 20000, 1000,0)
```

3.1 Background correction methods

FISHalyseR supports three different methods to correct for uneven illumination.

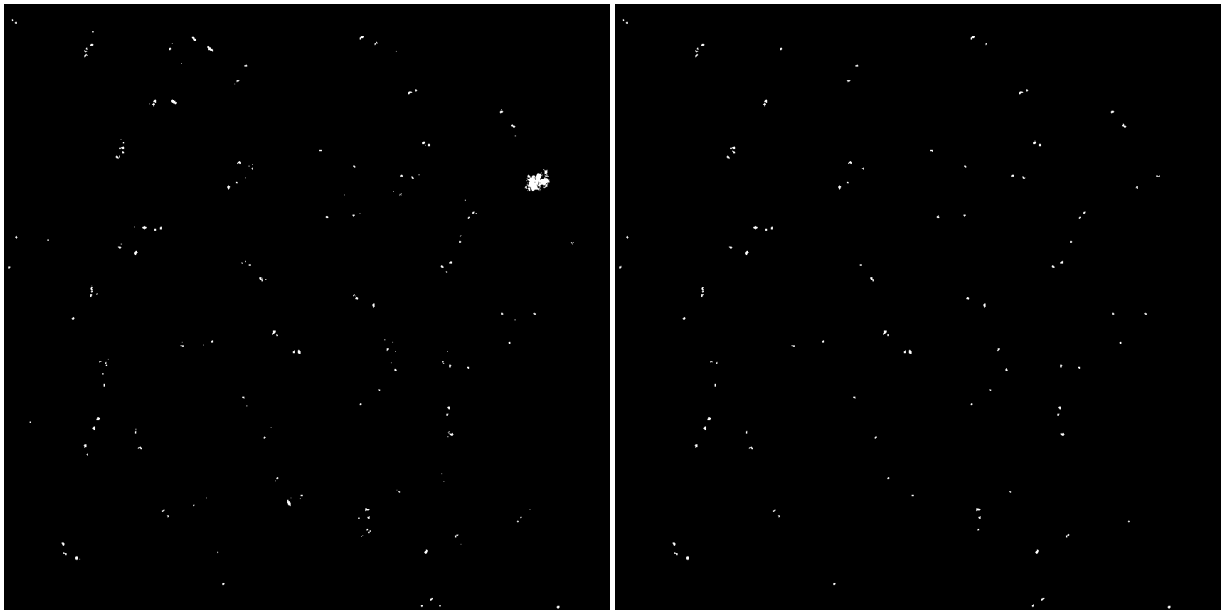

Figure 3. The original input image is given on the left side. Applying analyseParticles with MinSize=5 and MaxSize=20 results in the image shown on the right side. Components smaller than 5 pixels and larger than 20 pixels have been removed.

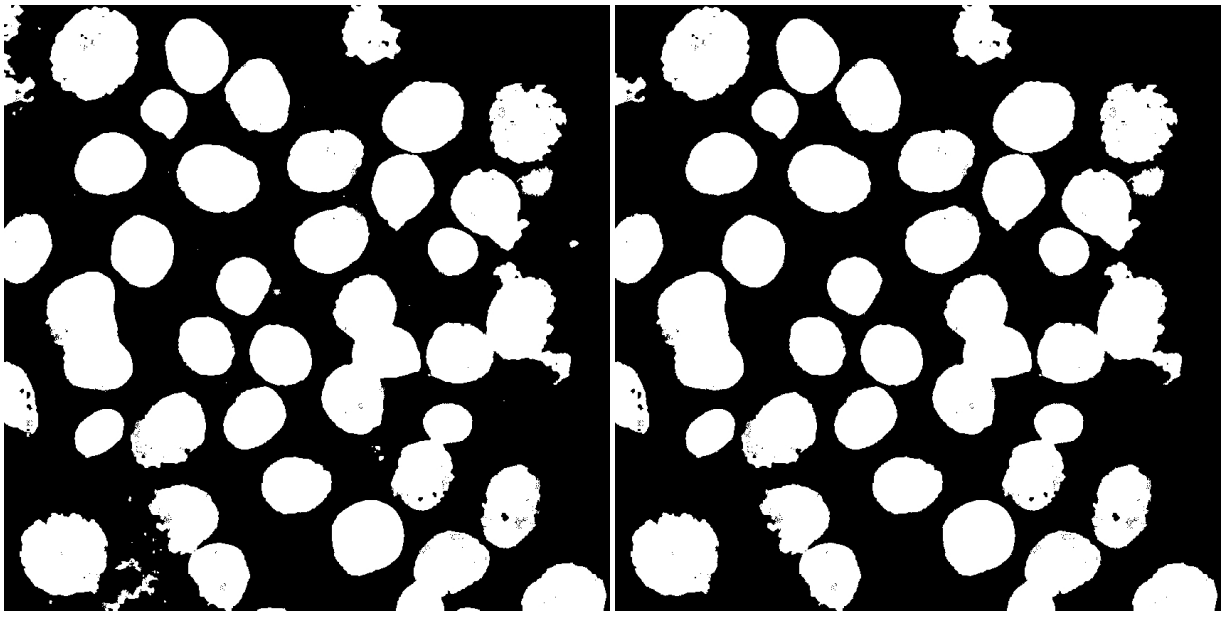

Figure 4. This figure illustrates the application of analyseParticles on the nuclei mask. The input image is shown on the left side. Components (nuclei, artefacts) with an area greater than 20000 pixel and smaller than 1000 pixels have been removed.

1. Gaussian blurring
2. User-specified illumination image
3. Multidimensional Illumination Correction using an image stack

To apply (multiple) Gaussian blurring, option 1 has to be chosen. The second parameter specifies the sigma.

```
bgCorrMethod = list(1,100)
```

In case the user has an illumination gradient image it can be passed to FISHalyseR using option 2.

```
# path to illumination image
illumPath = "../IllCorr.jpg"
bgCorrMethod = list(2,illumPath)
```

If multiple images from the same acquisition or the same microscopic setup are available then option 3 should be chosen. A novel multidimensional illumination correction method using a stack of images is then applied to derive the illumination gradient. Now the last parameter specifies the amount of images chosen to estimate the gradient. The more images are chosen the better the resulting illumination correction image will be. In the example beneath six images are chosen located in the given path.

```
bgCorrMethod = list(3,"/path/to/stack","*.jpg",6)
```

The variable of type list specifying the illumination correction method is later passed to `processFISH`.

3.2 Choosing probe channels

FISHalyseR currently supports two different ways to read probe channels. In case each probe has been acquired in a separate channel, simply pass each single file to FISHalyseR in a list variable. For visualisation purpose each channel has to be assigned a colour. This is achieved by creating a list variable containing the colour vectors for each channel.

```
red_Og   <- system.file( "extdata", "SampleFISH_R.jpg", package="FISHalyseR")
green_Gn <- system.file( "extdata", "SampleFISH_G.jpg", package="FISHalyseR")
#Create colour vector list
channelColours = list(R=c(255,0,0),G=c(0,255,0))
channelSignals = list(red_Og,green_Gn)
```

In case the user has only a composite image, thus all probe channels have been fused to a RGB image, then FISHalyseR separates the RGB image into single channels Red, Green, Blue. This input format will only work with a maximum of three probes because mixtures of red, green and blue can not be separated by simple colour channel splitting.

```
combinedImage <- system.file( "extdata", "SampleFISH.jpg", package="FISHalyseR")
channelSignals = list(combinedImage)
```

The variable of type list specifying where FISHalyseR can find the probe channels is later passed to `processFISH`.

4 Quantifying FISH probes in cell culture images

A single function `processFISH` has to be called to process a cell culture image with FISH probes. Note that the amount of probe channels is not limited assuming that sufficient main memory. Results are written to a CSV file. An image (Figure 5) is created illustrating the location of each nucleus (shown in cyan) as well as each detected probe (red, green in our example). Note that the list input parameters **bgCorrMethod** and **channelSignals** are described in detail in the previous paragraphs.

1. combinedImg - composite image, RGB images composed of all available stains
2. writedir - Output directory
3. bgCorrMethod - list with two arguments. Currently four options are available:
   (a) - None
   (b) - gaussian blur
   e.g. bgCorrMethod=list(1, 100)
   (c) - User-provided illumination correction image
   e.g. bgCorrMethod=list(2,"/exIllCorr.jpg")
   (d) - illumination correction if multiple images from the same acquisition are available
   e.g. bgCorrMethod=list(3,"/path/to/stack","*.jpg",6)
4. channelSignals - list of paths to image containing the probes
5. channelColours - colour vector for each channel (list type)
   e.g. channelColours=list(R=c(255,0,0),G=c(0,255,0),B=c(0,0,255))
6. sizeNucleus - c(5,100) - Analyse only nuclei within that range (in pixels)
7. sizeProbe - c(5,100) - Analyse only probes within that range (in pixels)
8. maxprobes - Maximum limit for probes per nuclei. Nuclei with more probes will be ignored
9. outputImageFormat - specify output format e.g. jpg or png

The example beneath demonstrate the usage of FISHalyseR. Only the `processFISH` functions has to be called. After specifying the signal channels, colour vectors, area constrains and maximum amount of probes to analyse, an output image is created per nucleus as well as a CSV file summarising all measurements.

```
illuCorrection = system.file( "extdata", "SampleFISHillu.jpg", package="FISHalyseR")

combinedImage <- system.file( "extdata", "SampleFISH.jpg", package="FISHalyseR")
red_Og   <- system.file( "extdata", "SampleFISH_R.jpg", package="FISHalyseR")
green_Gn <- system.file( "extdata", "SampleFISH_G.jpg", package="FISHalyseR")

# directory where all the files will be saved
writedir = paste(TempDir,sep='')

bgCorrMethod = list(0,'')

channelColours = list(R=c(255,0,0),G=c(0,255,0))
```

```r
channelSignals = list(red_Og,green_Gn)
sizecell = c(1000,20000)
sizeprobe= c(5,20)


processFISH(combinedImage,writedir,bgCorrMethod,channelSignals,
            channelColours,sizecell,sizeprobe)

## setting up directory structure
## No illumination correction method selected
## Extracting mask from the image....
## thresholding the nuclei ...
## analysing the nuclei ...
## Dividing nuclei
## Processing channel probes
## loading channels .....
## loading channel :  R
## loading channel :  G
## extracting features from the cell mask image
## saving label image with cell id
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
```

```
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## Warning in sum(w[z]):  integer overflow - use sum(as.numeric(.))
## computing features ....
## Processing channel R
## Processing channel G
## Computing distances between probes
## computing distance between R and R
## computing distance between R and G
## computing distance between G and G
## saving data to csv file ............
## processing cell id:  1
## processing cell id:  2
## processing cell id:  3
## processing cell id:  4
## processing cell id:  5
## processing cell id:  6
## processing cell id:  7
## processing cell id:  8
## processing cell id:  9
## processing cell id:  10
## processing cell id:  11
## processing cell id:  12
## processing cell id:  13
## processing cell id:  14
## processing cell id:  15
## processing cell id:  16
## processing cell id:  17
## processing cell id:  18
## processing cell id:  19
## processing cell id:  20
## processing cell id:  21
## processing cell id:  22
## processing cell id:  23
## processing cell id:  24
## processing cell id:  25
## processing cell id:  26
## processing cell id:  27
## processing cell id:  28
## processing cell id:  29
## processing cell id:  30
## processing cell id:  31
## processing cell id:  32
```

```
## processing cell id:  33
## processing cell id:  34
## processing cell id:  35
## processing cell id:  36
## processing cell id:  37
## processing cell id:  38
## processing cell id:  39
## processing cell id:  40
## processing cell id:  41
## processing cell id:  42
## processing cell id:  43
## processing cell id:  44
## Done processing fish data ...
```

Results are stored in a CSV file named after the input file. An example of an output with only one probe channel (red) is given beneath. In case of multiple channels the amount of columns will vary because then also distances between each different probe channel will be listed.

filename	Name of the input file
cell id	Unique number per nucleus
eccentricity	Eccentricity (shape feature)
number of R probes	Amount of red probes for a specific nucleus
R1 R2	Distance in pixels between red probe 1 and red probe 2
R1 R3	Distance in pixels between red probe 1 and red probe 3
R2 R3	Distance in pixels between red probe 2 and red probe 3
X center of mass	X coordinate of the center of mass of the nucleus
Y center of mass	Y coordinate of the center of mass of the nucleus
area of nucleus	Area in pixels of the nucleus
perimeter of cell	Perimeter of nucleus
radius of cell	Radius of nucleus
AR1	Area in pixels of red probe 1
AR2	Area in pixels of red probe 2
AR3	Area in pixels of red probe 3

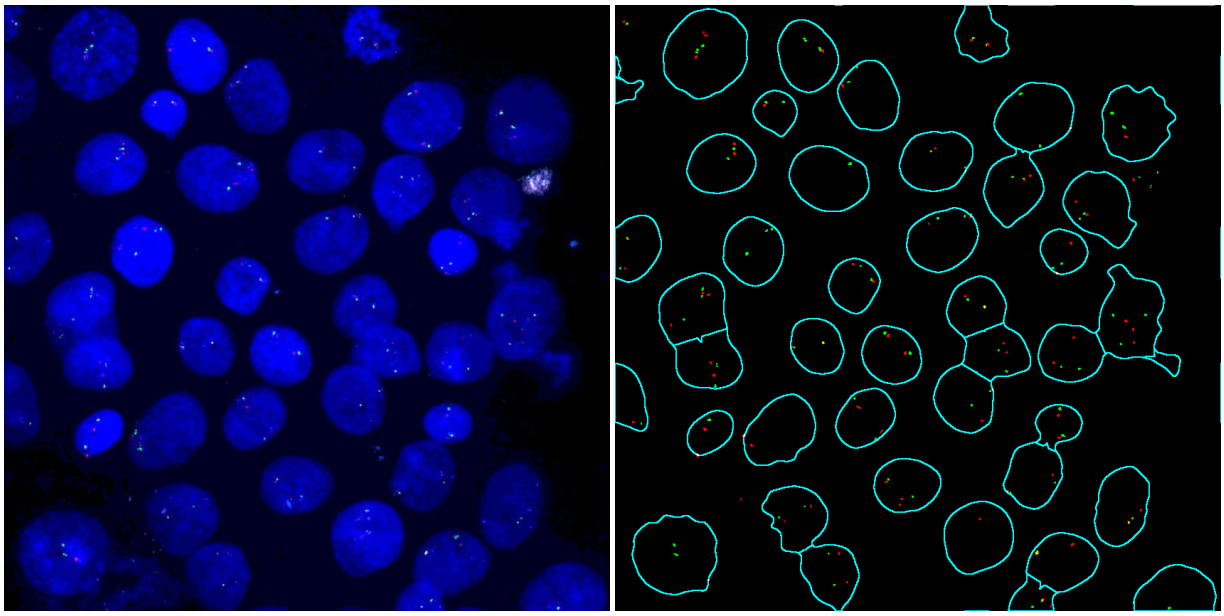

Figure 5. This figure shows the acquired composite image of a FISH culture next to resulting image of FISHalyseR. The outlines of each detected nuclei is depicted in cyan. Probes are drawn in their respective colour (red, green)