IT 10 027

Examensarbete 15 hp
June 2010

Nuclei segmentation on bright-field images

Francisco Cruz Fernández

Institutionen för informationsteknologi
Department of Information Technology

Abstract

Nuclei segmentation on bright-field images

Francisco Cruz Fernández

Nuclei segmentation is a common and complicated task in image analysis. There is no general solution for the problem, and depending on the image characteristics the segmentation can be performed in different ways. Bright-field images add some complications to the problem; the color of some elements of the image is close to the color of the nuclei, making the segmentation difficult. In this thesis some methods are presented to complete this task, two classifiers, minimum distance classifier and multilayer perceptron are tested to enhance the nuclei. After the classification, threshold methods together with morphological operations are used to get the segmentation of the nuclei with an accuracy around 85%.

Handledare: Amin Allalou
Ämnesgranskare: Amin Allalou
Examinator: Anders Jansson
IT 10 027
Tryckt av: Reprocentralen ITC

Contents

1. Introduction

1.1. Overview

The area of image analysis has had significant progress in recent years. Sophisticated algorithms and more powerful hardware have led to the development of new techniques that have revolutionized areas such as medical imaging. It is now possible to obtain images with a good precision from any part of the human body and manipulate them to obtain the desired information.

Nuclei segmentation is an important task since it can be used to perform diagnosis of tumor cells and other diseases. There are biological functions that appear to be related with the change in the geometry of the nuclei, and specific analysis of them is required to advance in the research. For these tasks, the area of image acquisition is well advanced with new microscopes and techniques, such as bright-field images we have available for this project.

Bright-field images are very common in the area of biomedical research. However perform automatic segmentation of these images is a challenge, especially finding the cell nuclei. Images in this project have signals with color close to the color of the nuclei, making the segmentation ever more difficult.

1.1. Project aim

The aim of this project is to get an overview of the existing methods of nuclei-segmentation both in bright-field and other images, and implement and try some of these methods for the images used in this project.

2. Background theory

2.1 Digital image analysis

In image analysis, we know a digital image as a set of pixels where each one of them has a concrete value forming the desired image. A pixel is the smallest part of a digital image that we can find and its intensity range is variable depending the image type. For volume images, the smallest parts are called voxels, and the shape is a cube.

There are different ways of represent an image depending of the range of the pixel values. In binary images, for example, there is only one bit for each pixel to represent the image, black or white. A higher number of bits provide more information into the image.

It is possible to see a digital image as a mathematical function of three variables f (x, y, b); where x and y represent the spatial variables to locate each pixel, and b represents the spectral band in a multispectral image. The most common way to represent color images is by the RGB model, formed by three spectral bands; red, green, and blue. Each channel consist of a grey-scale image, usually 8 bit representing $2^8 = 256$ different grey levels. The three grey-scale images form the RGB color image.

The treatment and manipulation of digital images is called digital image processing, image analysis is using image processing to retrieve information from images.

2.2. Previous work

Getting a good nuclei segmentation in medical images is a hard task, many methods have been developed to solve this task. A model-based segmentation for images with clumped nuclei is presented in [1]; it is a common problem to have nuclei clump together and it is necessary to find a way to separate one nucleus from another. The authors try to solve this problem using several methods such as thresholding, extraction of crease segments and polygonal approximation. In [2] Viterbi search-based dual active contour algorithm was used to get a good segmentation rate in a huge image databases. In [3] morphological watersheds and other operations are used to accomplish the same. In this article there is a lot of over-segmentation, from of direct application of watersheds to noisy images. This was solved with homotopy modification, which gave very good results.
In [4] a pattern recognition based segmentation system (PRS-system) was developed to classify nuclei from 138 images of tissues stained with hematoxilin and eosin stain (HE-stain). The system was designed to use one of these classifiers: Minimum distance, Bayesian, or a Multi-layer Perceptron (MLP). Useful conclusions on error rates and the usage of these classifiers are shown. Another technique used for this task is the usage of fuzzy logic to distinguish the nuclei from the background [5]. For that they use the membership grade (MD) and membership function (MF) together

with the selected nuclei features to deduce the result using a set of fuzzy rules.

There are many other ways to deal with this problem, e.g., clustering methods, edge-detection [6], region growing methods [7], specific neural networks (PCNNs), and a lot of morphology-based operations.
There are different ways of solving this task, depending on the cell culture dyed, image source or special features of the nuclei.

2.3 Used tools

2.3.1 MATLAB

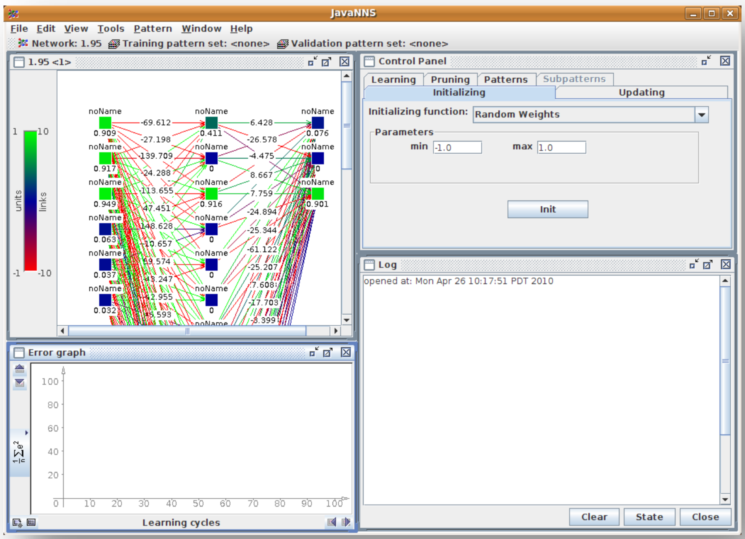
MATLAB, created in 1970 by Cleve Moler, is a numerical computing environment developed by The MathWorks. MATLAB is widely used in academic and professional field with more than a million users in the last year.
The environment allows matrix manipulations, thousands of predefined functions, implementation of algorithms and interfacing with other languages as C, C++ and Fortran. MATLAB also provides a great set of image analysis functions.

2.3.2 SNNS (Stuttgart Neural Network Simulator)

Stuttgart Neural Network Simulator (SNNS) is a software simulator for neural network originally developed at the University of Stuttgart. The aim of the SNNS project was to create an efficient and flexible simulation environment for research on and application of neural sets.

The SNNS simulator consists of a simulator kernel written in C and a graphical user interface for easy creation and simulation of the network. The software also provides the user with included network architectures and learning and transfer functions as Backpropagation (BP), Quick prop or Radial Networks among others.
After the success of SNNS in the field, a Java version called JavaNNS appears offering a new interface developed in Java with a powerful editor and visual tools to make easier task. Despite the advantages of this version, the most experienced users still use the first version due to its flexibility and the possibility of work with scripts over the networks.

For our task, we will use JavaNNS in its version for Linux (Fig.1).



Initializing function: Random Weights

Parameters

min -1.0 max 1.0

```
opened at: Mon Apr 26 10:17:51 PDT 2010
```

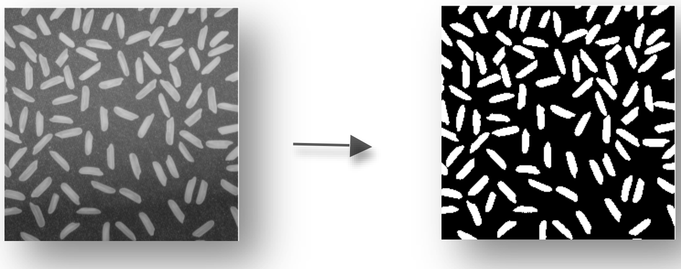
Figure 1: Main window of JavaNNS for Linux

2.4. Algorithms and operations

2.4.1 Segmentation

In image analysis, segmentation is the process of partitioning a digital image into multiple parts in order to analyze them more easily. This technique is used to locate and separate objects according to specific characteristics, getting the location of the objects either by its boundaries or by its form.

There is no general solution for segmenting an image, a lot of different techniques can be used, e.g., edge-detection [6], histogram-based methods, neural networks segmentation, etc. These techniques often have to be combined with morphological operations or other operations depending on the task.



(a) (b)

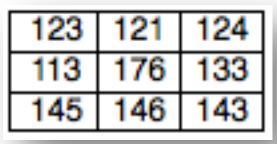
Figure 2: (a) Original Image. (b) Image after segmentation

2.4.2 Median filter

Median filtering is a nonlinear filtering technique used to reduce noise in an image. Filtering operations are often used in image analysis as first step of the main task, offering the chance of enhance the image and removing the existing noise.

Median filter acts over each pixel (x, y) of the image replacing its value f (x, y) with the median of its neighbors. For that it is necessary to specify a filter mask of size NxM specifying the number of neighbors used for the median calculation.

An example of the use of a 3x3 median filter is shown in the image below.



123	121	124
113	176	133
145	146	143

Figure 3: Selected pixel values

Neighborhood values:
113,121,123,124,133,145,146,143,176

Median value = 133

2.4.3 Minimum distance classifier

Minimum distance classifier is a method used to classify between classes using the distance concept, is one of the simplest classifiers but still powerful. Its operation is based on measuring the distance from the element to be classified to the class representatives and choose the nearest class as its own.

The classifier requires a distance function to calculate the distance between any pair of points in the representational space. One of the most used commonly distance is the Euclidean. To calculate the Euclidean distance between two points in a D-dimensional space the following formula (1) is used:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_d (x_d - y_d)^2} \qquad \text{for all } \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^D \tag{1}$$

The samples used to train the classifier must be labeled according to the class to which they belong (supervised learning). For each class the mean of all elements (in that class) is calculated.

Once the mean of all classes are calculated, an unclassified element (pixel) can be classified by finding the distance to all classes and assigning the element to the nearest class (2).

$$c(x) = \arg\min_c \ d(x, p_c) \qquad \text{for all } x \in E \tag{2}$$

In the formula, E represents the set of elements to classify, and P_c each representative of the class.

Figure 4 shows one hypothetical case of classification between two classes: ‘o’ and ‘+’. The mean of each class is represented as a filled element between the elements of the class and the element to classify is represented by θ. After calculate the distance to each class, the element will be classified as ‘o’.

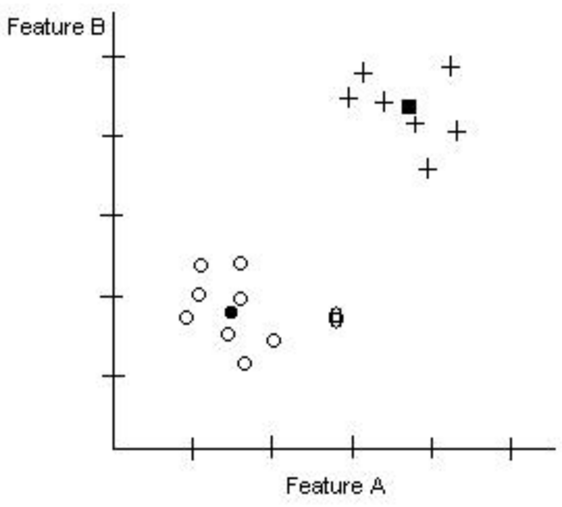


Figure 4: Representational space of the classification process.

2.4.4 Neural networks: Multilayer perceptron

An artificial neural network (ANN) is a mathematical model inspired in the operation way of the animal nervous system. It consists of an interconnected group of artificial neurons working together to produce an output stimulus.

In the network, each single neuron gets an input and emits an output given by three functions:

- A propagation function that generally consists in the summation of each input multiplied by the interconnection weight.
- An activation function that modifies the propagation function output according to the activation function type.
- A transfer function applied to the activation function output delimiting it according to the interpretation of the desired output. Some of the most common ones are the sigmoid function, values between [0,1], and the hyperbolic tangent, values between [-1,1]

A multilayer perceptron is an ANN consisting of multiple layers that allows solving non-linear separable problems (the main limitation of the single perceptron, which can only solve linear-separable problems). The layers can be classified into three types:

- Input layer: This layer contains the input neurons, which receive the input patterns of the network. In a network, there is only one input layer and there is not processing in the neurons of this layer.

- Hidden layer: This type of layer represents the core of a neural network. The information from the input layers is processed through the hidden neurons calculating the correct weights for the smooth running of the network. Multilayer perceptron is characterized by having one or more hidden layers, and depending of the problem the configuration of these layers must be carefully studied.

- Output layer: The output values of the neurons of this layer are the output for the whole network and also the classification result.

In a multilayer perceptron, the output of the layer *i* is connected with the input of the following neurons from the layer *i+1* as seen below:

The learning stage on a multilayer perceptron consists in the modification of the weights between nodes according to the input values of the network.

One of the most used learning functions is backpropagation. The algorithm calculates the error in the output nodes according to the target values from the input and the produced values by the perceptron, and then modifies the network weights according to this error. After several training steps the error reaches a minimum and the training has to be stopped.

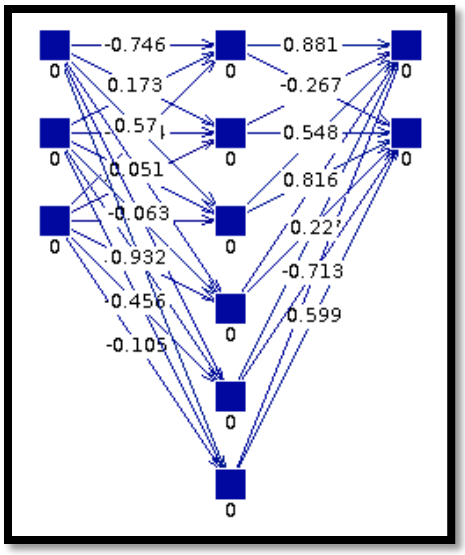


Figure 5: Weights between layers on a neural network

Once the network is trained and the weights fixed, the classification process will be performed using the input values of the element to classify with the network and analyzing the output values.

In case of bad results after train the network some changes may be considered to improve the working, e.g., changing the network configuration (number of neurons in hidden layer and number of hidden layers), the initial weights, the parameter of the learning function or even vary the format of the samples and the data sets could be useful.

2.4.5 Threshold (Otsu's method)

Thresholding is the most common and among simplest method used in image segmentation. As all segmentation methods, the main idea is to assign every pixel into a certain group and that is what Otsu's method does.

The algorithm assumes that the image contains two groups of pixels, background and foreground pixels, and calculates the optimum threshold level to separate them so that the dispersion within each segment is as small as possible, but at the same time the dispersion is a high as possible between different segments. This value is obtained when the quotient between both variances is the maximum possible.

Once the threshold value is calculated, the method decides if one pixel belongs to one certain group comparing its gray level with the calculated threshold level.

$$T_{global}(g) = \begin{cases} 0 & \text{si } g < t \\ 1 & \text{si } g \geq t \end{cases} \quad (3)$$

The process works according the previous formula (3). One pixel of the image is marked as '1' if its value is greater than the threshold value, and as '0' if not. Pixels marked as '1' (white) belongs to the object, and the rest of elements are considered as background.

There are several variants of this method better suited to certain types of images. Sometimes a global threshold level for the whole image is not a good idea is there are areas of the image with very different intensity values or more illuminated than other. In this case, global thresholding will produce a bad result segmenting more elements from the darkest area than from the clearer one.

- Local thresholding

The first variation is called local thresholding. With this method the image is divided into regions and a threshold value is calculated for each one of them. In this case, the segmentation of each region will be independent of the other regions, thus the differences in lighting or other will not affect the final result.

$$T_{local}(x,y) = \begin{cases} 0 & \text{si } g(x,y) < t_i \\ 1 & \text{si } g(x,y) \geq t_i \end{cases} \quad \forall (x,y) \in \text{Region } R_i \quad (4)$$

The main disadvantage of this variant is that the boundaries between the regions are visible in the result. This happens when the threshold levels of adjacent regions are very different producing this undesired effect.

- Dynamic thresholding

Another way to deal with the global Otsu's method problem avoiding undesired effects is calculating the threshold level for each pixel according to a defined window. This method is an extension of local thresholding. For each pixel the threshold is calculated from an NxN window around the pixel.

$$T_d(x,y) = \begin{cases} 0 & \text{si } g(x,y) < t(N(x,y)) \\ 1 & \text{si } g(x,y) \geq t(N(x,y)) \end{cases} \quad (5)$$

Thus, t(N(x, y)) represents the threshold of the selected pixel in the region N. This variation is very stable against illumination changes, however the required computational effort in this case is very high, and this method must be used only in specific cases when other methods do not produce good results, or computational time is not an issue.

2.4.6 Bilinear interpolation

Bilinear interpolation is a method of constructing new data points within the range of a set of known data points in two dimensions.
This process is used when we want to find the value of an unknown function f at one point (x, y) knowing the value of this function in other four points Q_{11}, Q_{12}, Q_{21}, Q_{22}. It is possible to estimate the values from the region formed by these four points as is seen in the image on the right.
The formula to calculate the value of one point into the region is:

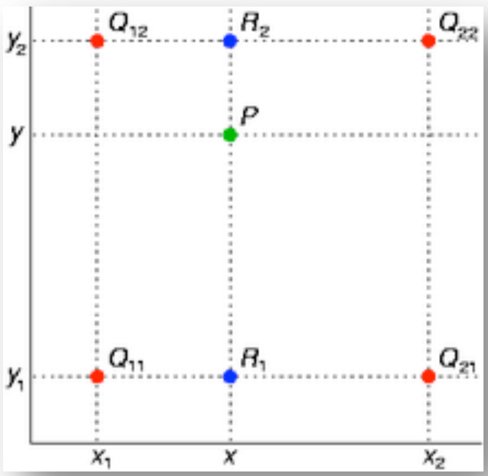


Figure 6: Q_{ij} represent the known points. P is the point to interpolate. See [10].

$$\begin{aligned} f(x,y) \approx\ & \frac{f(Q_{11})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y_2-y) \\ &+ \frac{f(Q_{21})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(y_2-y) \\ &+ \frac{f(Q_{12})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y-y_1) \\ &+ \frac{f(Q_{22})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(y-y_1). \end{aligned} \quad (6)$$

2.4.7 Morphological operations

In image processing morphological operations are used to change the shape or the appearance of an object in an image. Most of them are usually used in segmentation in order to get a better definition of the objects.

- Filling

Filling operations are used to fill small holes of certain shapes. This operation is often used to complete the region of the elements after an operation that could damage them. In the image on the right after a pixel-classification process some central pixels were not well classified getting small holes inside of them. Filling operations are used in these cases to repair the region of the objects.

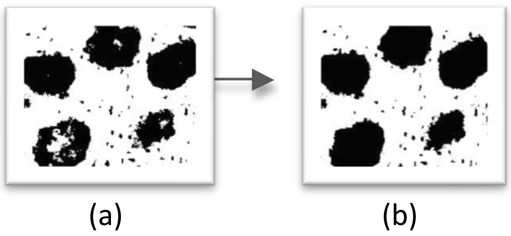


Figure 7: (a) Nuclei with holes before filling. (b) Nuclei after filling

- Opening operation

Opening and closing operations are used to remove morphological noise in images. Opening removes small objects and smooth the shape of the elements, while closing fill small holes from the elements and joins little objects with near elements.

An opening operation is an erosion followed by a dilation. These operations smooth the contours through a size modification. Erosion removes small and thin elements, and dilation fill small holes and small entries. The combination of both operations allows noise removal and shape smoothing without changing the size of the objects.

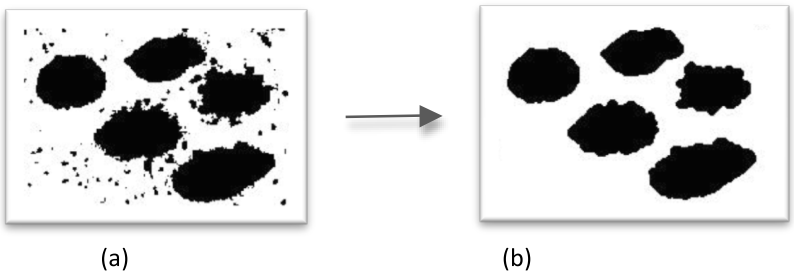


Figure 8: (a) Nuclei before opening operation. (b) Nuclei after opening operation

3. Materials and methods

3.1 Image acquisition

The images used in this project are similar to the images used in [8]. In this work, bright-field images were confirmed as an important tool for basic and clinical research.

Bright-field images were acquired with a Leica DFC320 camera over a Leica DMRE microscope. The tissue samples were previously treated in the laboratory and stained with horseradish peroxidase (HRP), a common enzyme widely used in biochemistry due to its ability to amplify weak signals and increase detectability of target molecule.

3.2 Image set

For this thesis we used 27 tissue images. The images have a resolution of 2304x3072 that add up to 7.077.888 pixels for each channel red, green and blue. We can realize that due to the large number of pixels in one image, the algorithms may take a long time to compute the results, something to keep in mind for the following operations and methods.

In the images it is possible to distinguish different elements with a naked eye. The nuclei can be seen as the purple-blue circular forms, the red dots are stained proteins, and the white is the background. The interest areas are the nuclei, but we can see that the red dots may be problematic if we try to threshold the image without pre-processing. Thats why we need to try to enhance the contrast of the nuclei and remove artifacts added by the red dots into the image.

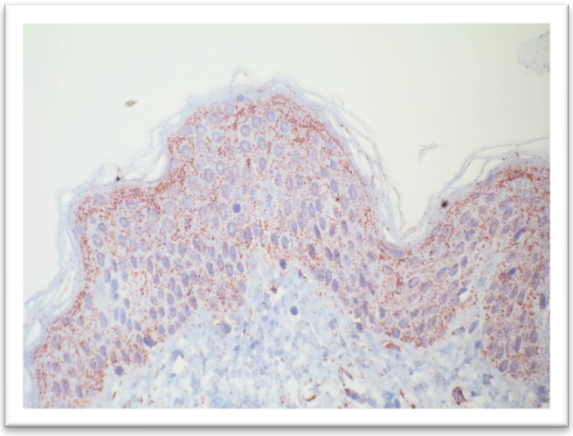


Figure 9: Image from the set

Another special feature to keep in mind about the image is the illumination. In all images dark region on the right is present. This will affect the final result of our algorithms, as will be seen later.

The following images belong to the red and green channel from one image in the dataset; it is possible to see the role of the red dots darkening the image and overlapping the nuclei.

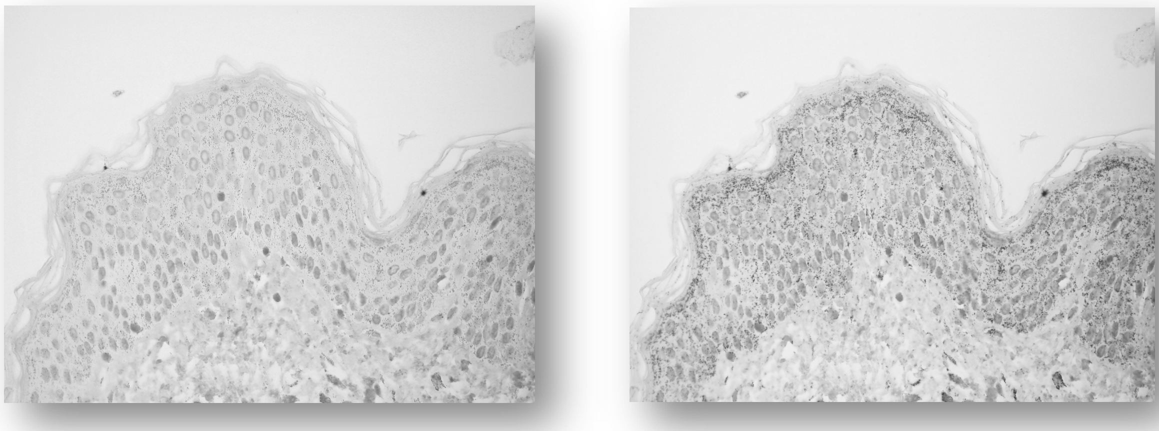

(a) (b)

Figure 10: (a) Red channel from the image on figure 8. (b) Green channel from the image.

3.3 Classifying nuclei by minimum distance

The most common first step working with digital images is to try to enhance the image, and removing noise. In this case, some defined filters and operations on MATLAB were tested looking for a better definition of the nuclei shape and a reduction of the noise introduced by other tissue elements. In most cases, nuclei shapes were altered by the direct application of filters, but good result was obtained with the median filter. A 10x10 mask was used for this filter producing a noise reduction and removal of the red dots. There was no other tested method that offered significant improvements of the image.

Once the image is pre-processed, minimum distance classifier was chosen first to try to obtain a better contrast of the nuclei.

The classifier will work on a 3-dimensional space given by the three channels RGB. 100 nuclei samples were manually selected to prepare the classifier, each one of them as the intensity of one pixel belonging to one nucleus on each RGB channel. The mean of those samples was calculated, getting the nuclei class identifier. For the calculations of the distance between elements the Euclidean distance was used.

The classification process using minimum distance is usually obtained by calculating the distances to each class and choosing the minimum. Instead of that, as there is only one class to discriminate, the distance to the nuclei class was used.

For each pixel from the original image, the distance between its position on the 3-dimensional space and the position of the nuclei class identifier was calculated. This distance is a number close to zero for the pixels belonging to the nuclei class, and higher for the pixels belonging to other areas. Printing those values into a new image, nuclei intensity should be lower than the intensity of the other elements, getting dark areas as nuclei and brighter areas for the rest.

A visual representation may help to understand the process. In the image below the nuclei samples are printed in blue, the mean of them in red, and the green point belongs to one random pixel of the image. The classifier will calculate its distance from the green to the red point, setting this value as the intensity for that pixel in the result image.

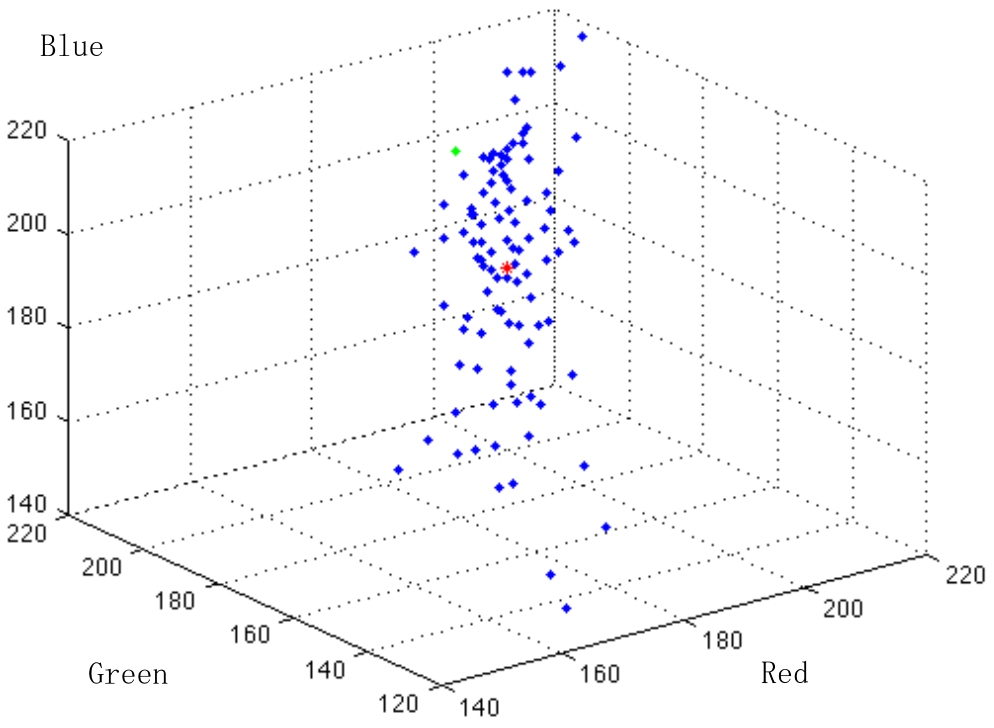


Figure 11: Representation of the samples and the point to classify in the 3-dimensional space.

The results after the classification over five different images were not as good as expected. The illumination problem of the images and the different intensities between nuclei produced bad results.

Details about the result are explained in section 4.

3.4 Multilayer perceptron classification

After the use of minimum distance another more complex classifier was tested. The chosen classifier for that was the multilayer perceptron, a type of neural network formed by multiple layers that allows to solve problems like this. The expected result in this case is similar to the previous case, remove the noise produced by the red dots and other elements and enhance the nuclei color in order to get a good segmentation.

Classifier data set

The first step on the classifier design is to choose the structure of the samples, number of classes to discriminate and number of samples for each step of the classification.

- Classes: The classifier will be designed to discriminate between 3 different classes. This classes will be: nuclei (1), red dots (2), and background (3).

- Samples: 5750 samples were manually selected from the original RGB image; 2750 of them belong to the nuclei, 1500 to the red dots, and 1500 to the background. Each sample will represent a region of 6x6 pixels region from the corresponding class.

- Features: 6 features were selected for each sample, the average of the 36 pixels into each RGB channel and the standard deviation between the same pixels. Therefore the structure of one sample in MATLAB code will be like thus:

  ```
  Sample(i)=[meanR   meanG   meanB  stdR  stdG   stdB  C]
  ```

  The variable C represents the class to which the sample belongs, being 1,2, or 3.

- Normalization: For the smooth running of the classifier is recommended to normalize the network inputs. To do that, each element of the sample will be divided by the maximum of them plus one in order to fit into the interval [0 , 1]. For instance, the normalization of the meanR values in MATLAB code will be:

  ```
  samples(:,1) = samples(:,1)/(max(samples(:,1))+1)
  ```

Network construction and configuration tests

To build the network and perform the training and classification processes we used the neural network simulator JavaNNS presented in section 2.3.2.

First of all it is necessary to define the size of the training set, validation and test for the learning process. For that, the whole sample set was shuffled and divided into the three required sets; this process was repeated twice choosing different set sizes to compare the results between the two combinations of the data set.

- Data set 1

Training set = 4000 samples	(2500 nuclei, 750 red dots, 750 background)
Validation set = 750 samples	(400 nuclei, 175 red dots, 175 background)
Test set = 750 samples	(600 nuclei, 200 red dots, 200 background)

- Data set 2

Training set = 3600 samples	(2000 nuclei, 800 red dots, 800 background)
Validation set = 600 samples	(300 nuclei, 150 red dots, 150 background)
Test set = 650 samples	(300 nuclei, 175 red dots, 175 background)

Once the data set is created the next step is the selection of the network configuration, which means choose the number of neurons for each layer of the network. The number of features of the samples gives the number of neurons to the input layer, and the number of classes fixes the number of the output neurons. Knowing this only the number of hidden layers and its size have to be chosen.

For the training stage *Backpropagation* was chosen as learning function. For each network configuration values of the learning factor from 0.1 to 0.7 were tested. The parameter *dmax* of the learning function was fixed to 0.

For the first data set different configurations have been tested; the graph below shows the mean square error (MSE) for each number of neurons using one hidden layer. The value shown belongs to the minimum MSE obtained for all the tested learning factors.

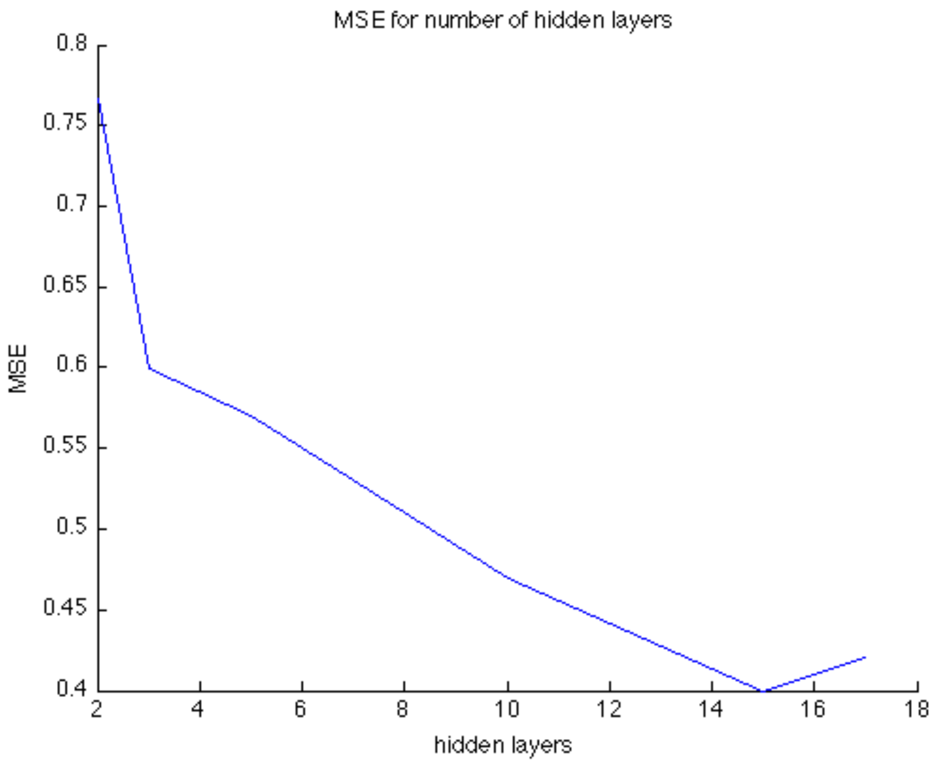


Figure 12: MSE value for number of hidden layers for the data set 1.

The minimum MSE was obtained with 15 hidden neurons in the hidden layer with a learning factor of 0.7, getting an error rate of 2,5% on the test set.

After that, different configurations with two hidden layers were tested on the same way. The MSE value for these configurations was similar and in some cases lower, however the training stage was much slower than with one hidden layer and the error rate after the training with the test set was higher in comparison with the error rate with 15 hidden neurons in one hidden layer. For this reason the configurations with 2 hidden layers was discarded.

The same experiment was performed with the second data set. The main difference of this data set is that the number of nuclei samples was reduced in respect to the number of red dot samples or background samples. An excessive number of samples of one class may lead to an overfitting, where the network instead to learn about general patterns, is focused in the individual examples. The graph below shows the MSE for each configuration tested using the second data set.

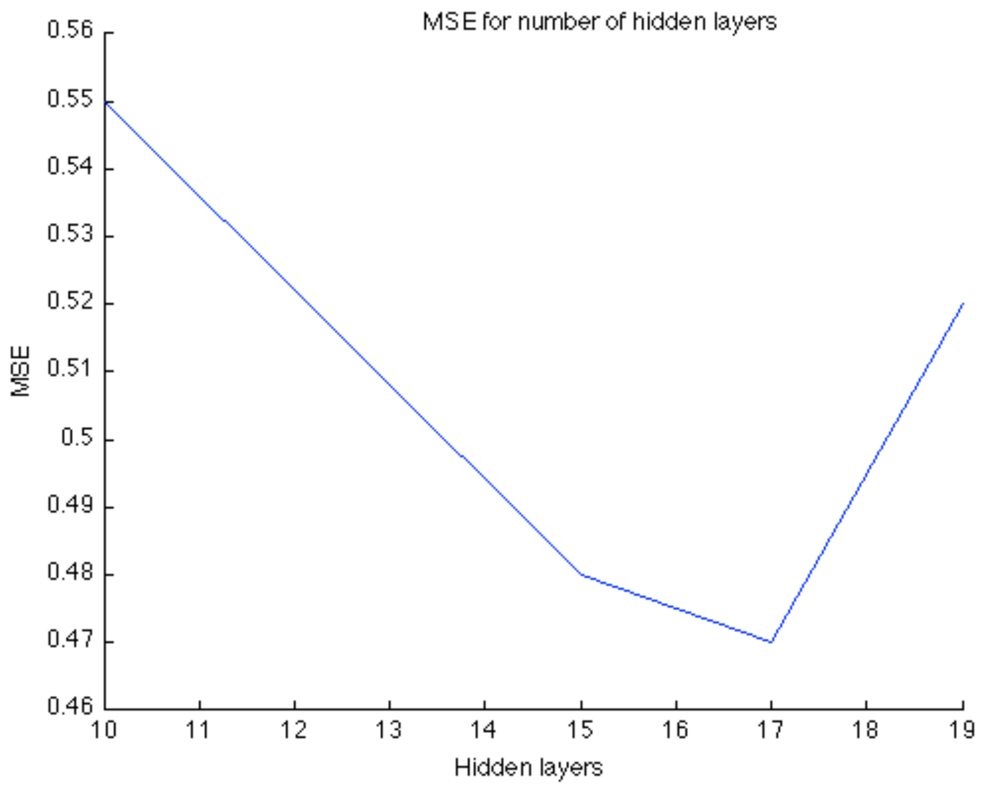


Figure 13: MSE value for number of hidden layers for the data set 2.

In this case, the minimum MSE was obtained for 17 hidden neurons in the hidden layer with a learning factor of 0.3, getting an error rate of 1.95% on the test set.

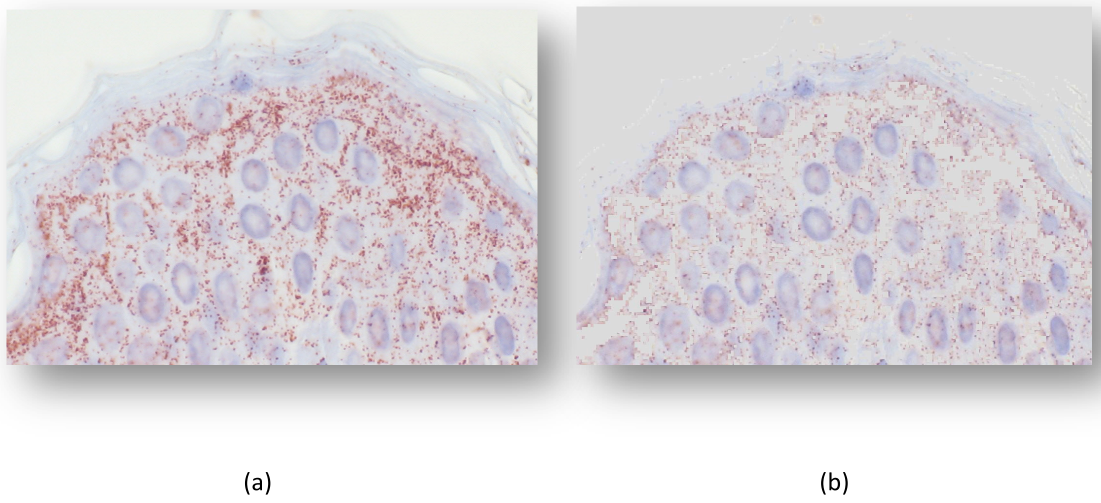
After testing the different network configurations, the chosen configuration was the network with 6 neurons on the input layer, 17 on the hidden layer, and 3 on the output layer, trained with Backpropagation using a learning factor of 0.3.

Classification

The next step after the network is created is to classify the images. The input values for the network must have the same structure that the samples have, that means it is necessary to process the images before using the network.

The process is the same that to get the samples, but in this case instead of do it over concrete areas, the process will affects to the whole image. One image has 196.608 areas of 6x6 pixels, therefore the codified image will be represented by 196.608 lines where each line contain the 6 features before commented for each area.
Once the image is processed the network can read it and classify. The classification result will be one number between 1 and 3 for each 6x6 area from the image; this number will represent the class to which it belongs.

The final result will be a new RGB image where the classified areas belonging to the class 2 and 3 have not been printed, reducing considerably the noise from the red dots and from other elements of the image. The classification of one image from the image set is shown below.



(a) (b)

Figure 14: (a) Part from the original image. (b) Same part after classification with neural network.

3.5 Segmentation by threshold

Once a great part of the image noise was removed, it is possible to see the nuclei shape well defined making possible for a better segmentation.

The chosen technique to do that was thresholding by Otsu's method. In the first place Otsu's method was applied calculating the threshold level from the whole image. As was said in section 3.2, there are differences between the illuminations on different parts of the images; this produced a threshold level that was too high for the clearer areas and too low for the darker areas. This resulted in an image where some nuclei are almost deleted, and others were unrecognizable under a black stain.

The first solution to deal with this problem was to use the local version of Otsu's method. For that, the image was divided into 9 equal regions and Otsu's method was applied for each one of them individually. In this case the result was better than in the previous case. Despite the good results obtained with local threshold there was an added problem to solve; the boundaries of each region are reflected in the image result producing an undesired effect as it is shown below.

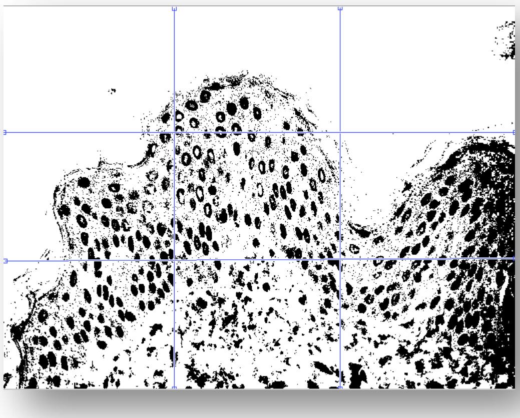
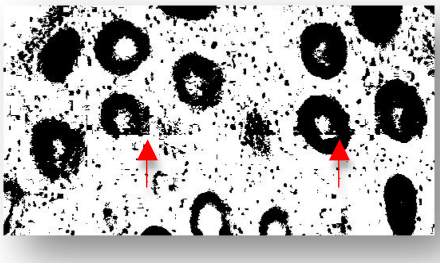



Figure 15: The image on the left shows the selected regions for local thresholding. On the right is shown the result after local threshold, where the region boundaries are reflected.

The first way to try to solve this problem was using dynamic threshold. This algorithm calculates the threshold level for each pixel over an NxN window. The expectations were good with this method, however the time to process the whole image was too long and another alternative method had to be used.

Bilinear interpolation of thresholds was chosen to solve these problems. Threshold values for each region used in local thresholding will be used for this task. Center pixels of each region will be fixed with the calculated threshold value for each one; these points will define a new grid formed by 4 regions over which the interpolation will be performed.

The first step of the interpolation will calculate the values for the pixels into the blue square in the image on the left. Once these values are calculated, the same process will be repeated to interpolate the rest of the image. In this case the required fixed values to perform the interpolation will be calculated as a lineal combination from the values used on the first step.

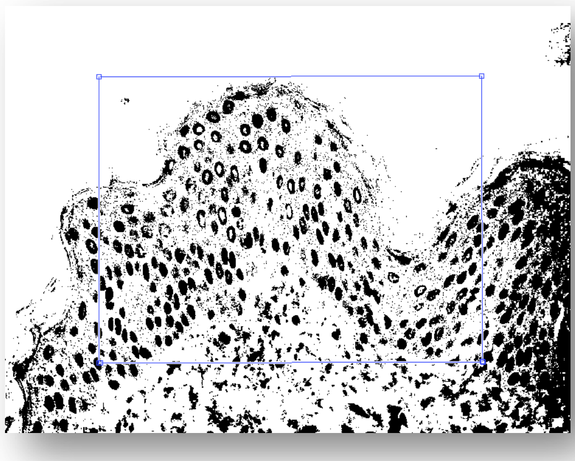
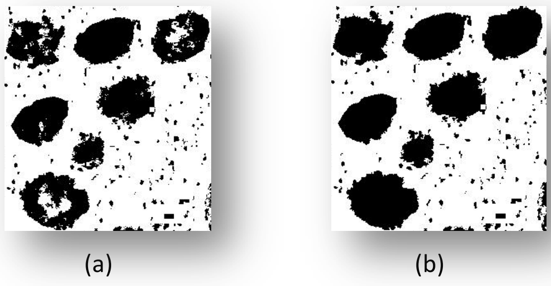


Figure 16: The blue square indicates the first interpolated area.

The results after bilinear interpolation of threshold values save better results than the other methods; despite of this not only the nuclei were segmented, some noise is still damaging the image and it is necessary to use morphological operations to remove these elements.

3.6 Morphological operations

Defined operations provided by MATLAB will be used to perform this stage; at this point the image had some elements that needs to be remove in order to isolate the nuclei and get a good segmentation.
The first problem is about the holes in the nuclei; most nuclei do not have the same intensity in the boundaries than in the center, because of this after the threshold operation some nuclei have holes inside them that need to be filled. To solve this was used the *imfill* function in MATLAB on the result after thresholding. This operation is shown below.



(a) (b)

Figure 17: (a) Nuclei before filling. (b) Nuclei after filling.

After filling the holes, some nuclei were still clumped together. To separate them and remove small objects an opening operation was used with disk radius 4.

Once the nuclei were filled and the noise was removed, the next step is to find which of the segmented elements are nuclei and which are not. To do that the function *regionprops* from MATLAB was used. This function calculates several parameters for each element of the image such as area, orientation, perimeter, etc. In this case the most relevant parameters to discriminate between elements were the area and the eccentricity of each element.
After calculate the area an eccentricity of each element, the next step is to remove smaller and bigger elements assuming that they are not nuclei. The chosen limit to discriminate between them was an area under 900 and over 15000. These values were chosen analyzing the area from 50 known nuclei and nuclei clumps getting an approximation of the maximum and minimum area of them.

The same process was performed about the eccentricity; the elements with an eccentricity over 0.97 were removed from the image, leaving the nuclei with only some elements that could not be detected.

4. Results

In this section the results for every tested method will be shown. First, the image result after classification by minimum distance will be presented, in this image is possible to see the illumination problem and the different intensities of the nuclei. In second place one of the RGB images obtained by classification with neural networks will be shown and compared with the original one in order to see the noise reduction between them. The third point of this section is a comparison between the three different threshold methods used. Finally, the final segmentation after morphological operations will be presented and overlapped with the original image showing the result of the segmentation.

- **Minimum distance results**

The image result after classification by minimum distance is shown below. The dark area on the right of the image belongs to the bad illuminated area; this characteristic appears in the whole set of images and may be due to a fail in the microscope illumination or some another error in the acquisition. This condition leads to a general darkening at some areas making it hard to find the nuclei.
Another visible problem is about the intensity of some nuclei clearer than the rest. These nuclei are classified with intensity similar to background elements loosing a great number of them.

This result belongs to the classification of the image shown in section 3.2.

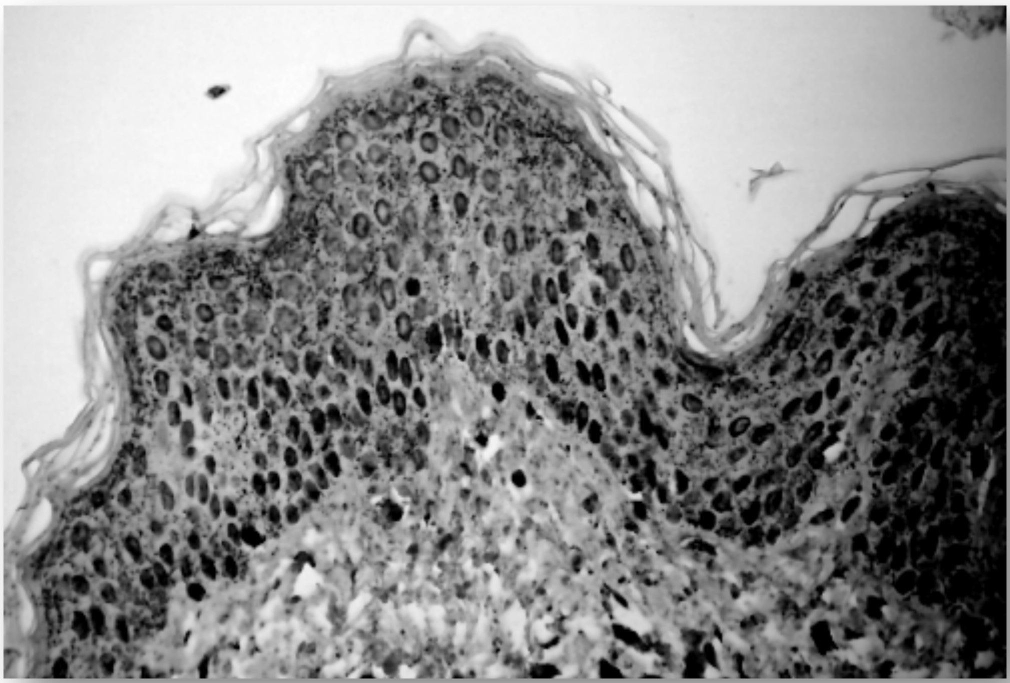


Figure 18: Image result after classification by minimum distance.

- **Multilayer perceptron results**

After the image classification using a neural network type multilayer perceptron good results were obtained. Each 6x6 area classified as class 1 or 2 was removed from the original image producing an important noise reduction both in red dots and background elements.
Without noise reduction segmentation by threshold would have been impossible; red dots and the nuclei would have linked each other forming clumps making the nuclei recognition a very hard task.
Despite the good results some elements are still perturbing the nuclei. Tissue boundaries and some blue areas as on the left of the image have similar color-intensity than the nuclei and it is difficult to distinguish between them at least with the current selected features.
A possible way to improve the result is the selection of other features for the samples; in this case the information provided by the images is not rich enough to extract good features, but according to the obtained results it is true that with more information from the images to calculate features the final classification could be better. Nevertheless the obtained results are considered acceptable for this task.

The classification of one image from the image set is shown below.

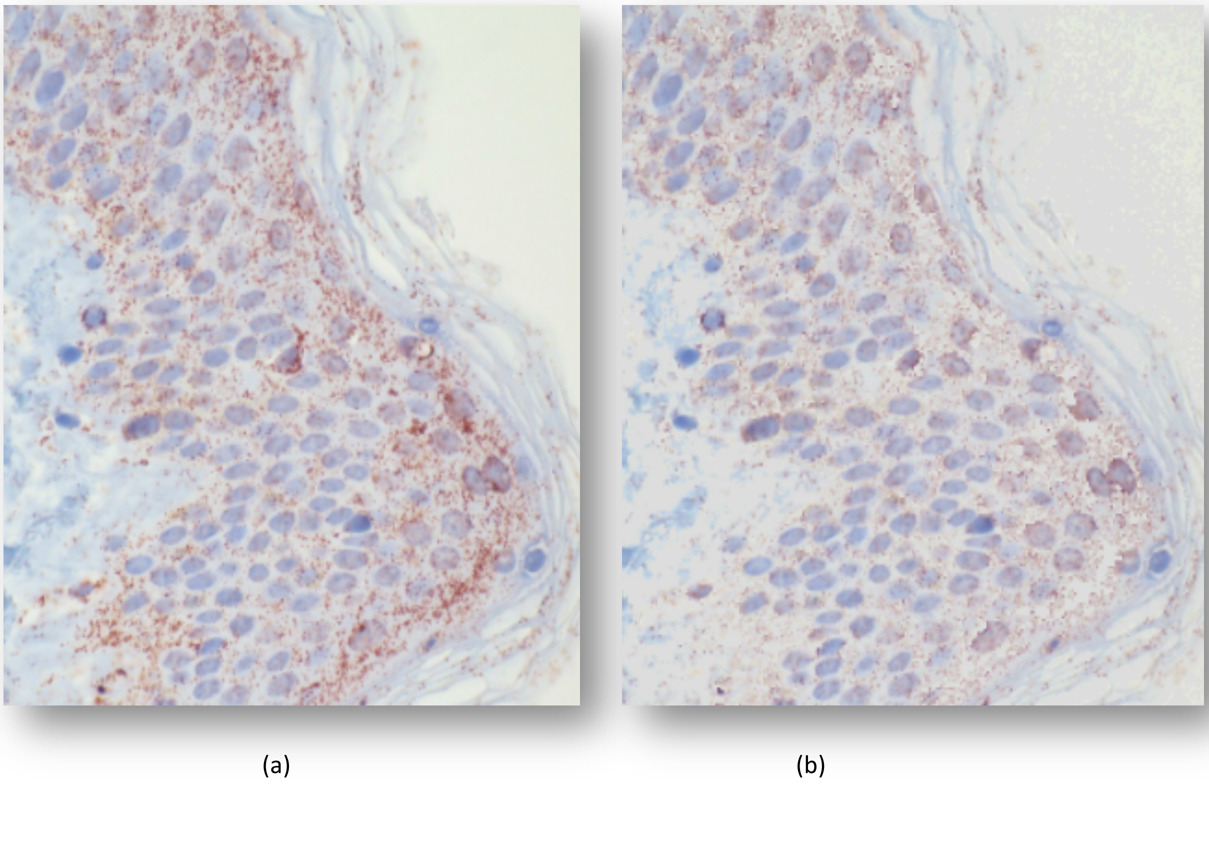
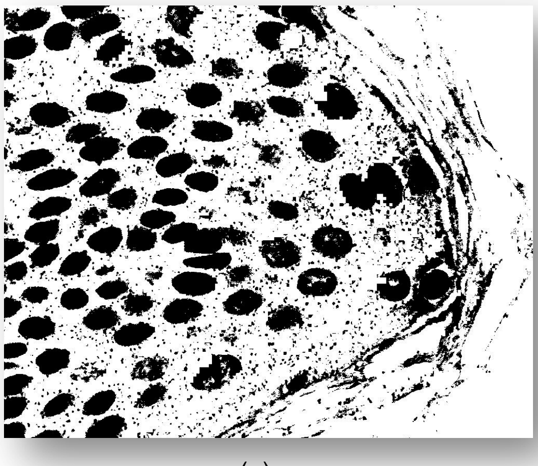


Figure 19: (a) Part from the original image. (b) The same image after classification.

- **Threshold results**

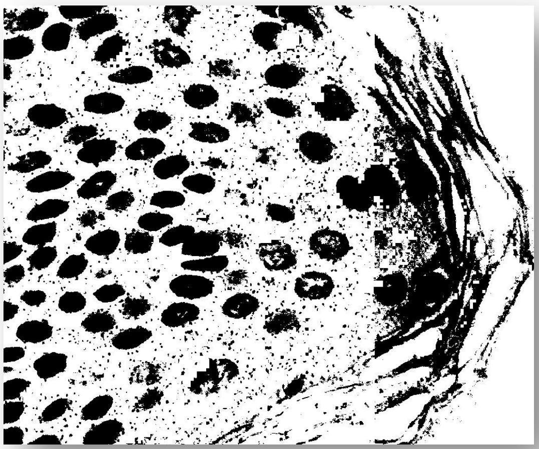
- Global threshold

The results after global thresholding would have been considered as good if the illumination problem had not existed. This problem together with the different concentration and intensities of the nuclei caused a noise increase in some areas and the disappearance of some nuclei in other. Some nuclei were joined due to the noise increase forming clumps making the segmentation fail.


(a)

- Local threshold

The previous problems were solved using local threshold, however some new problems appeared. When the difference between the threshold levels of adjacent regions was high, the boundaries were seen in the image result producing an undesired effect as is seen in the image on the right.
With this method more holes appeared into the nuclei due to the lower intensity on the center, however this problem was easily corrected later with morphological operations.


(b)

- Bilinear Interpolation of thresholds

Bilinear interpolation of thresholds was the best solution to this problem. After this process the nuclei were well identified even with the illumination problem. This result was chosen as a final method to use.

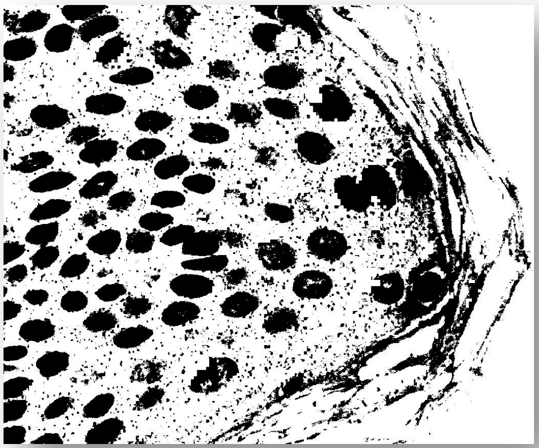

(c)

Figure 20: (a) Result after global threshold. (b) Result after local threshold with the reflected boundary. (c) Final result of bilinear interpolation.

- **Morphological operations result.**

The image on the right shows the final result after the application of all morphological operations.
After the application of filling and opening operations noise belonging to the little elements on the image was removed. Due to the opening operation most nuclei clumps were separated and its boundary was smooth.
Despite of the nuclei separation some of them still remained joined forming elements with a big area. One of the consequences of filtering elements by shape was the elimination of some of these nuclei clumps since they were considered as background elements or as strange shapes.
There are some elements that are not nuclei and were not removed after the filtering, however these elements represent a minority and do not affect the final result.

After these operations a high percentage of segmented nuclei was obtained and considered as a good result.

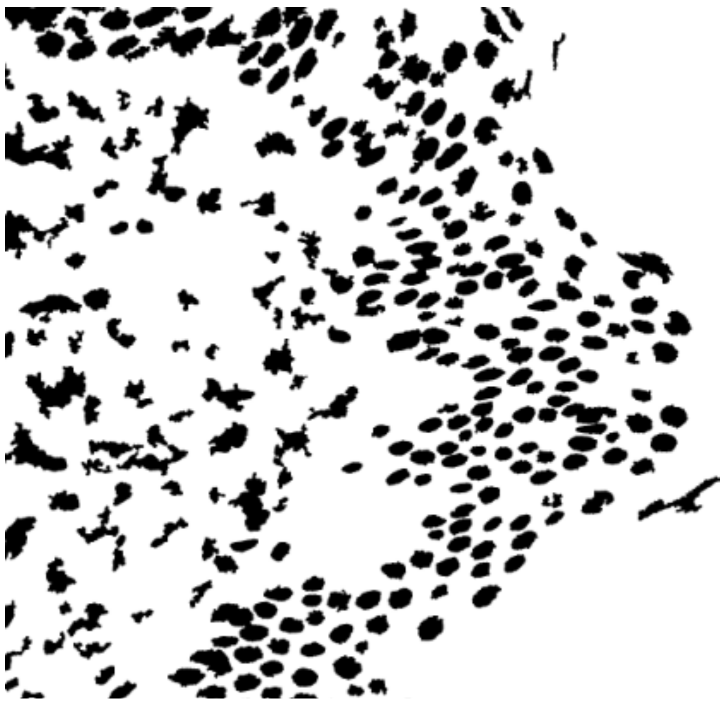


Figure 21: Nuclei aspect after morphological operations.

- **Computational cost**

Work with big images implies a high computational cost on some of the used methods. The use of some methods had to be avoided due the large number of pixels in the images, however in this case the results were obtained in a reasonable time. The task was performed on a Core2Duo at 2.26 GHz giving a segmentation time for one image of 175 seconds.

- **Final result**

The following image shows the final result of segmentation superimposed on the original image. An average of 85% correctly segmented nuclei, compared to manually detecting the nuclei, was obtained on two images in the dataset.

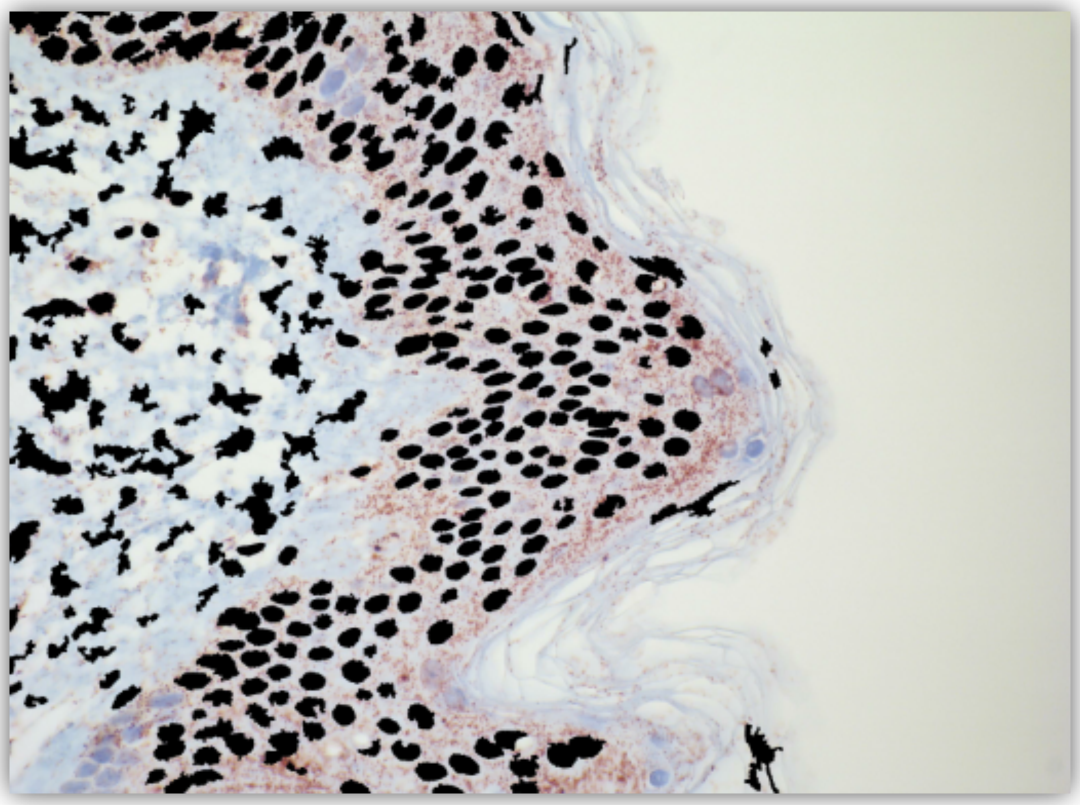


Figure 22: Segmentation result superimposed on the original image.

5. Conclusions

Nuclei segmentation in bright-field images is a complicated task due to low contrast between the elements of the image. The attempt of classify the nuclei by minimum distance did not produce the expected results due the illumination problem of the images and the few information provided by them. Multilayer perceptron was chosen to perform this task, proving that a more sophisticated classifier may produce good results on this task. The classification result provided a great noise reduction that allowed for a good threshold.

The differences between the intensities of the nuclei from different areas were demonstrated during the threshold. Global and local threshold produced defects in the image result, forcing to use bilinear interpolation of thresholds to produce a good threshold of the nuclei.

Complementary operations to enhance the nuclei shape and isolate them from the rest of elements are used in most works about nuclei segmentation. In this case, filling operation was used to fill the body of the nuclei, and several more operations were used to remove the remaining elements. Despite this, several thin forms and similar to the nuclei could not be removed, but these did not affect the final result, that much.

Furthermore, work with big images implies a high computational cost on some of the used methods, and had to be considered before applying the chosen methods.

6. Future work

Focusing on this work there are some improvements that could be done to improve the final segmentation. The results of the MLP classification could be improved with another representation of the samples. Find special features for each class could lead to a better classification. There are many types of neural networks and learning functions, the use of other types could also improve the results.

On the other hand, the use of advanced morphological operations could avoid segmenting objects that are not nuclei, obtaining a better final result.

6. Acknowledgments

I would like especially to thank my supervisor Amin Allalou for all his help and support during the time I have been working with him. Also, I want to thank to everyone in the CBA for receiving me these months, and especially thanks to the Uppsala Universitet together with the Universidad Politécnica de Valencia for providing me with the opportunity of spend this year in this wonderful country.

8. References

[1] G. Cong, B. Parvin. “Model-based segmentation of nuclei”.

[2] P. Bamford, B. Lovell. “Unsupervised cell nucleus segmentation with active contours”. *Signal Processing* 71 (1998) 203-213.

[3] Jose Alfredo F.Costa, Nelson D. A. Mascarenhas, Márcio L. de Andrade Netto. “Cell nuclei segmentation in noisy images using morphological watersheds”.

[4] P. Spyridonos, D. Glotsos, D. Covouras, P. Ravazoula, V. Zolota and G. Nikiforidis. “Pattern recognition based segmentation method of cell nuclei in tussue section analysis”.

[5] G. Begelman, E. Gur, E. Rivlin, M. Rudzsky, and Z. Zalevsky. “Cell nuclei segmentation using fuzzy logic engine”.

[6] M. P. Pathegama and Ö. Göl. “Edge-end pixel extraction for edge-based image segmentation”. *World Academy of Science, Engineering and Technology* 2, 2005.

[7] Kwang-Back kim, Doo Heon Song and Young Woon Woo. “Nucleus segmentation and recognition of uterine cervical Pap-Smears”.

[8] A. Zieba, C. Wählby, F. Hjelm, L. Jordan, J. Berg, U. Landegren, and K. Pardali. “Bright-Field microscopy visualization of proteins and protein complexes by In Situ Proximity ligation with Peroxidase detection”. Clinical *Chemistry* 56:1 (2010).

[9] A. Allalou, “Algorithm design for signal detection in fluorescence microscopy image of cells”.

[10] http://en.wikipedia.org/wiki/Bilinear_interpolation

[11] http://en.wikipedia.org/wiki/Segmentation_%28image_processing%29

[12] http://en.wikipedia.org/wiki/Otsu%27s_method

[13] http://en.wikipedia.org/wiki/Multilayer_perceptron

[14] http://homepages.inf.ed.ac.uk/rbf/HIPR2/classify.htm