UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

IMAGE ANALYSIS AND CLASSIFICATION TECHNIQUES FOR LEISHMANIASIS DETECTION

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Sofia Melissa Limon Jacques

In partial fulfilment

of the requirements for the degree in

Audiovisual Systems ENGINEERING

Advisor: Elisa Sayrol

Barcelona, October 2017

Abstract

Leishmaniosis is considered a neglected disease that causes thousands of deaths annually in some countries, specially tropical and subtropical countries. It is caused by protozoa of the genus Leishmania spp., which develop their life cycle between a vertebrate host and an invertebrate vector that transmits the disease. There are various techniques to diagnose leishmaniosis of which manual microscopy is considered to be the standard. There is a need for the development of automatic techniques that are able to detect leishmania parasites in a robust and unsupervised manner.

In this document we present and compare two different procedures for automatizing the detection process. The first one uses conventional image processing methods that have been around for some time but are known to be robust in the field. The second one is linked to a more recent and evolving technology called deep learning, and has proven to deliver outstanding results.

Resum

La leishmaniosi es considera una malaltia desatesa que causa milers de morts anuals en alguns països, especialment en països tropicals i subtropicals. És causada pel protozou del gènere Leishmània spp., que desenvolupen el seu cicle vital entre un hoste vertebrat i un vector invertebrat que transmet la malaltia. Hi ha diverses tècniques per diagnosticar la leishmaniosi, de les quals la microscòpia manual es considera estàndard. Es necessita el desenvolupament de tècniques automàtiques capaços de detectar el paràsit leishmaniosi d'una manera robusta i no supervisada.

En aquest document presentem i comparem dos procediments diferents per automatitzar el procés de detecció. El primer utilitza mètodes convencionals de processament d'imatges que no són nous, però que s'han declarat com una eina robusta en el camp de la medicina. El segon està relacionat amb una tecnologia més recent i en evolució basada en tècniques d'aprenentatge profund (Deep Learning), les quals s'estan proclamant com a eines revolucionàries del món de la visió per a computadors.

Resumen

La leishmaniosis se considera una enfermedad desatendida que causa miles de muertes anuales en algunos países, especialmente tropicales y subtropicales. Es causada por protozoos del género Leishmania spp., que desarrollan su ciclo de vida entre un huésped vertebrado y un vector invertebrado que transmite la enfermedad. Existen varias técnicas para diagnosticar la leishmaniosis de las que la microscopía manual se considera el estándar. Existe la necesidad de desarrollar técnicas automáticas que sean capaces de detectar el parásito de leishmaniosis de una manera robusta y sin supervisión.

En este documento presentamos y comparamos dos procedimientos diferentes para automatizar el proceso de detección. El primero utiliza métodos de procesamiento de imágenes convencionales que no son nuevos, pero que han demostrado ser robustos en el campo de la medicina. El segundo está vinculado a una tecnología más reciente y en evolución vinculada a aprendizaje profundo (Deep Learning), que se ha declarado como revolucionaria en el mundo de visión para computadores.

Acknowledgements

First of all, I would like to express my sincere gratitude to Elisa Sayrol who gave me the opportunity to participate in this project, and for advising me and guiding me during these past months. It has been difficult at times, so thank you for sticking around.

I am also thankful to Albert Aparicio who has been an important contributor in the success of this project and has helped me to keep pushing myself to do better.

I would also like to mention a special thanks to Phillippe Salembier, for inspiring me in his classes and awakening my interest for image processing techniques. He is one of those teachers “you remember”.

And finally, I would like to thank my family and friends that have encouraged me during these five years of my degree. But specially to my parents and my little brother, who have always been there for me and helped me in times of doubt.

Revision history and approval record

Revision	Date	Purpose
0	10/09/2017	Document Creation
1	09/10/2017	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Sofia Melissa Limon Jacques	sofia.melissa.limon@alu-upc.edu
Elisa Sayrol	elisa.sayrol@upc.edu

Written by: Sofia Melissa Limon Jacques		Reviewed and approved by: Elisa Sayrol	
Date	08/10/2017	Date	09/10/2017
Name	Sofia	Name	Elisa
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables:

1. Introduction

Leishmaniasis is a disease caused by parasites of the Leishmania type, and infections in humans can be caused by over 20 different species. Next to malaria, Leishmaniasis is the second worst known parasitic killing disease; an estimated 700.000 to 1 million new cases and 20.000 to 30.000 deaths occur each year according to the World Health Organization [1]. The disease is transmitted to humans by the bite of infected female phlebotomine sandflies (over 90 sepcies are known) and is mainly found in regions where factors such as poverty, malnutrition, deforestation, and urbanization are present.

There are 3 main forms of the disease: Visceral leishmaniasis (VL), Cutaneous leishmaniasis (CL) and Mucocutaneous leishmaniasis.

Visceral leishmaniasis, also known as kala-azar, is the most serious form of the disease and is fatal if left untreated in over 95% of cases. There are a series of laboratory tests that can confirm VL diagnosis [2]. The most commonly used tests detect antibodies, which are markers that the body produces after being infected by VL. Currently, the best available diagnosis tool is the rK39 rapid test. It has many advantages as it is easy to perform, quick and cheap with results in around 10–20 minutes and US$ 1 per test, it can be used in remote places and does not require any special equipment or training. However, one of the major drawbacks of these tests is the fact that they cannot be used to detect relapse cases because antibodies remain present long after clinical cure [3] [4].

For this reason, direct observation of the Leishmaniasis parasite body in the microscopic image taken from bone marrow samples can be considered the gold standard for diagnosis. Nevertheless, it requires of technical expertise, and because of the quantity of steps required in manual diagnosis, this analytic technique is tedious and inclined to human mistake even in experienced hands, leading to possibly late conclusions and mistaken diagnosis.

The purpose of this project is to propose an alternative to manual observation and counting of the leishmania parasites present in the bloodstream, as this process is hard and time consuming, by creating software that automatizes the detection procedure through computer vision. We will be researching two of the main fields in cell segmentation and parasite identification and comparing their results.

One of the incentives for starting this project, in part, was a master thesis [5] that was carried out in 2014 by Jaume Fernàndez García about deep learning techniques for malaria detection in medical images. However, it is not a continuation of this project as the parasite images from both infections do not resemble.

The project can be divided into two distinctive parts. The first half of the project consists in developing software with more traditional image processing techniques that have shown to be efficient in other fields of medicine. The second half of the project was developed using relatively novel techniques that involve a machine learning process. Albert Aparicio, who from January to July worked as an intern researcher in the image processing department, was the main contributor to the success of this part, as he researched the best method and adapted it to our needs.

During the development of this thesis we have been in contact with the Escola Superior d'Agricultura de Barcelona who have provided us with the image database, Professor

Daniel Codina and particularly with Berta Raventòs Roca who did her final thesis on researching in vitro cultivation methods for leishmania parasites [6].

1.1. Requirements and specifications

The initial requirements needed to develop this project involve the following:

- A basic knowledge on image processing techniques, as this is the core of the project.
- An in depth research about the Leishmaniasis disease, regarding on how to detect the parasites in order to correctly differentiate infected from non-infected images.
- A thorough research in state of the art technology is key for developing innovative and improved software; cell segmentation and object recognition were the main terms of research, both in the fields of "conventional" image processing techniques and deep learning methods.
- A dataset with good quality images.

1.2. Work plan

1.2.1. Work packages

<table>
<tr><td>Project: Documentation</td><td colspan="2">WP ref: 1</td></tr>
<tr><td>Major constituent: Research</td><td colspan="2">Sheet 1 of 6</td></tr>
<tr><td rowspan="2">Short description:
This section involves researching the state of the art techniques for Leishmaniosis parasite detection. Furthermore, state of the art segmentation techniques, object detection techniques and morphological transformations will also be researched.</td><td colspan="2">Planned start date:20/02/2017
Planned end date:15/03/2017</td></tr>
<tr><td colspan="2">Start event:20/02/2017
End event:13/03/2017</td></tr>
<tr><td>Internal task T1: Research Leishmaniasis automatic detection state of the art techniques.
Internal task T2: Research how other techniques can be applied to identify leishmaniasis parasites.
Internal task T3: Project proposal and workplan.</td><td>Deliverable/s:
Project proposal and workplan</td><td>Dates:
05/03/2017</td></tr>
</table>

<table>
<tr><td>Project: Software development</td><td colspan="2">WP ref: 2</td></tr>
<tr><td>Major constituent: Pre-processing & segmenting</td><td colspan="2">Sheet 2 of 6</td></tr>
<tr><td rowspan="2">Short description:
To obtain the most precision when detecting parasites, a pre-processing can be applied to the image to enhance the details or to eliminate non-wanted objects. Then each possible parasite has to be segmented from the background in order to examine it individually later on.</td><td colspan="2">Planned start date:16/03/2017
Planned end date:16/04/2017</td></tr>
<tr><td colspan="2">Start event:14/03/2017
End event:27/04/2017</td></tr>
<tr><td>Internal task T1: Implementation
Internal task T2: Testing with different settings to find the optimal parameters.</td><td>Deliverables:</td><td>Dates:</td></tr>
</table>

Project: Software development	WP ref: 3	
Major constituent: Parasite identification	Sheet 3 of 6	
Short description: For the image being processed, we have two types of object to examine: intra-cellular and extra-cellular parasites. Individual examination is necessary to determine whether the identified objects are parasites or not Finally the total amount of parasites should be counted.	Planned start date:17/04/2017 Planned end date:30/06/2017 Start event:04/05/2017 End event:	
Internal task T1: Cell examination, amastigote detection and identification. Internal task T2: Promastigote parasite identification. Internal task T3: Number of parasites present in the image.	Deliverables:	Dates:

Project: Project critical review	WP ref: 4	
Major constituent: Document generation	Sheet 4 of 6	
Short description: Write the critical review of the project, mentioning the state of the project, if the proposed plan has been carried out correctly, if there have been any problems or delays, etc.	Planned start date:03/05/2017 Planned end date:07/05/2017 Start event:04/05/2017 End event:09/05/2017	
Internal task T1: Document writing	Deliverables:	Dates:

Project: Improvements	WP ref: 5	
Major constituent: Research	Sheet 5 of 6	
Short description: Once the software has been completed, other techniques may be researched, such as deep learning, to find ways of increasing the detection results.	Planned start date:1/07/2017 Planned end date:31/07/2017 Start event: 15/06/2017 End event: 31/08/2017	
Internal task T1: Research other techniques Internal task T2: Study deep learning techniques, concretely convolutional neural networks.	Deliverables:	Dates:

Project: Improvements	WP ref: 6	
Major constituent: Implementation	Sheet 6 of 7	
Short description: Research intern Alerbert Aparicio adapted a Convolutional Neural Network to resolve our problem, and my task was to understand the functioning of the network and to try it out with different parameters.	Planned start date:1/08/2017 Planned end date:31/08/2017 Start event:1/08/2017 End event: 02/10/2017	
Internal task T1: Study the python scripts implementing the CNN Internal task T2: Test network with different parameters	Deliverables:	Dates:

Project: Project report	WP ref: 7	
Major constituent: Document generation	Sheet 7 of 7	
Short description: The objective is to generate a document containing all of the research performed, and an in-depth explanation of the generated software.	Planned start date:01/09/2017 Planned end date:01/10/2017	
	Start event: 11/09/2017 End event: 08/10/2017	
Internal task T1: Document writing	Deliverables:	Dates:

1.2.2. Milestones

WP#	Task#	Short title	Milestone / deliverable	Date (week)
1	3	Project proposal and workplan	Deliverable	05/03/2017
2		Complete the segmentation	Milestone	08/04/2017
3		Software prototype	Milestone	30/06/2017
4	1	Project critical review	Deliverable	07/05/2017
5	2	Learn about deep learning	Milestone	15/07/2017
7	1	Final report	Milestone	30/09/2017

1.2.3. Gantt diagram

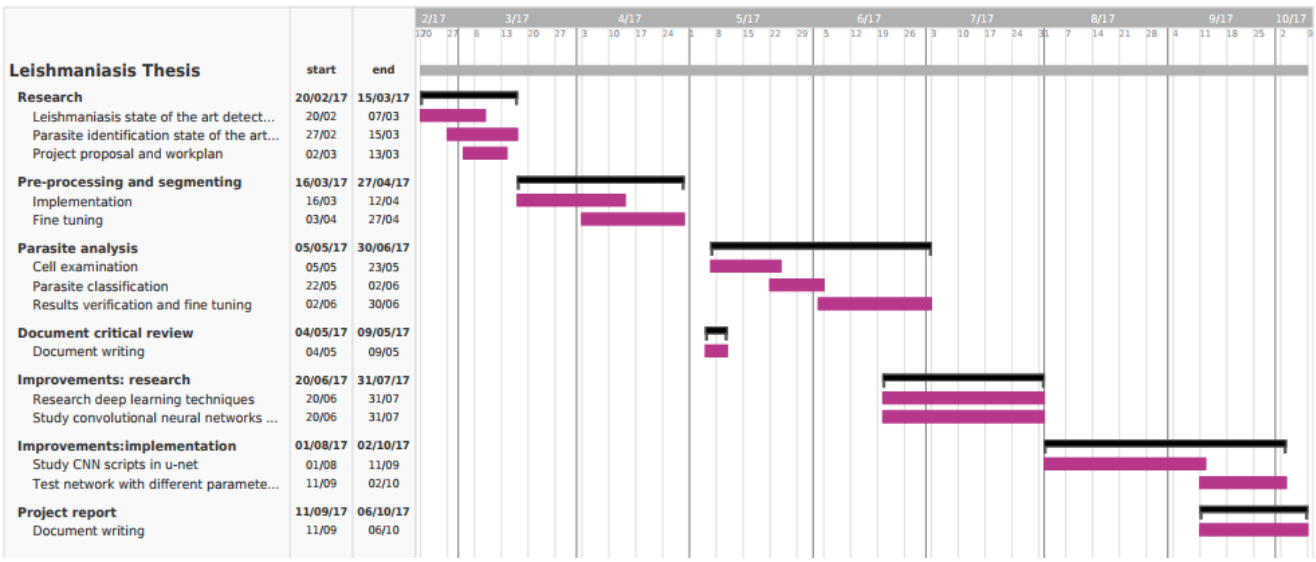

Figure 1.1. Gantt diagram

1.2.4. Incidences and deviations from the initial plans

The main setback during the development of the project was due to the limited resource of images to work with. Initially we were given a set of images of different microscopic magnifications with a lot of noise, and as a result it was hard to characterize the edges of the objects through computer vision techniques. Moreover, in some of the images the cells and the parasites were at an advanced phase of reproduction, making it hard to distinguish the boundaries and to identify the parasites.

In the second set of images, the quality was dramatically improved (as seen in **Figure 1.2**) and the colours present in the images were easier to work with, but they were not handed to us until early June. The parasites and the cells could be quickly identified through colour analysis making it easier for image processing. However, this second batch was taken with a mobile phone: the images consisted of a black background where the microscopic image was a circle region in the centre of the image; therefore, the area of interest had to be hand selected and cut before using them.

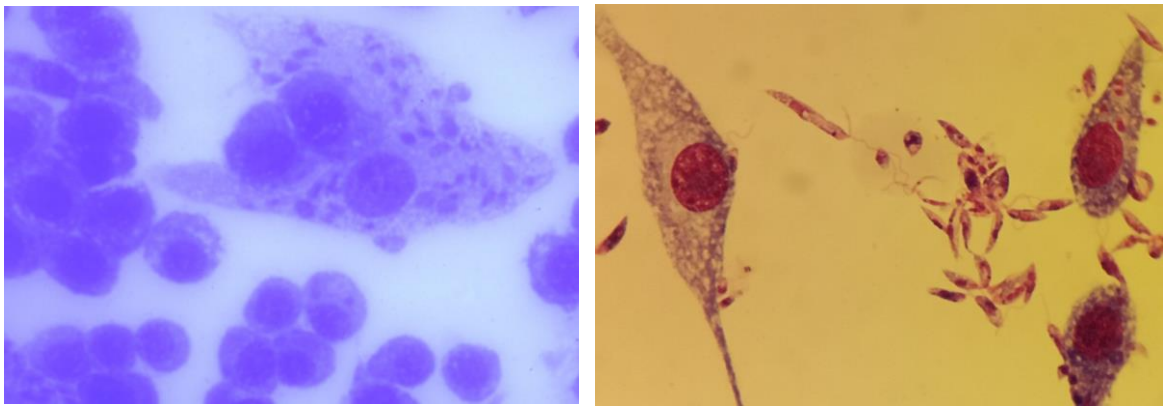

Figure 1.2. Initial images (left), second set of images (right)

In regards to the modifications to the initially planned calendar, the biggest change has been postponing the final delivery date from the end of June to the beginning of October. This fact has allowed to further research the field of deep learning techniques, implemented in the second half of the project.

2. State of the art of the technology used or applied in this thesis:

This section is intended for describing the basic techniques that have been used during the course of the project.

First we will describe methods that have conventionally been used to deal with image segmentation problems and object detection, and then we will proceed to explain the recently new deep learning techniques, concretely, Convolutional Neural Networks (CNN). The latter has proven to achieve outstanding results in image classification and segmentation field; since 2012 when a CNN entered the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [7] and beat by a 10.8% point margin (41% better than the next best) [8], CNNs have been the winning architecture in the ILSVRC and main focus of research.

With conventional methods, you fine tune a model with a specific set of parameters to perform a specific task, that is, to detect and recognize a certain object with certain features. However, deep learning techniques involve a learning of the machine as it learns to "train itself" to find the best parameters for the model given the input data.

The benefits of each of these methods can also be their restrictions. While classical image processing may perform good, new models have to be implemented every time the input data changes. For example, if we build a model to detect human faces, it is very likely to fail to recognize animal faces, or if the patterns in the dataset change, the parameters also have to be modified to fit the new data. On the other hand, even though deep learning methods can train the model for a variety of operations, a large dataset is needed in order to obtain reasonable results since a lack of data can often lead to overfitting, meaning that the model will only work for the training dataset, and in many cases the datasets are a limited resource.

2.1. Mathematical Morphology

Mathematical morphology (MM) is a nonlinear branch of the signal processing field and concerns the application of theoretical concepts to image analysis. The subject arose in 1964 and it is associated with the names of Georges Matheron and Jean Serra, who developed its main concepts and tools [9], presented it in several books and created a team at the Centre de Morphologie Mathematique on the Fontainebleau site of the Paris School of Mines. These operations are widely used to solve complex imaging applications involving shape recognition, image enhancement, image compression, image segmentation and image analysis, amongst others.

Morphology itself refers to the study of shapes and structures from a general scientific perspective. Therefore, mathematical morphology comprises a set of nonlinear transformations which modify the geometric features of an image. It was originally developed for binary images, and was later extended to grayscale functions and images.

The basic idea in binary morphology is to examine an image with a simple, pre-defined shape, drawing conclusions on how this shape fits or misses the shapes in the image. This simple shape is called the structuring element, and is itself a binary image (i.e., a subset of

the space or grid). The two fundamental operations in MM are erosion and dilation, from which you can derive the rest of operations.

These morphological operators take two pieces of data as input. One is the input image and the other is the structuring element (SE), which goes to every pixel and produces an output image based on a comparison between the input image and the SE.

From this point onwards, when we refer to foreground pixels we will be talking about the white pixels in binary images or bright pixels in greyscale images, and when we make a reference to background pixels we will be talking about black or dark pictures in binary or greyscale images.

In the following paragraphs, we will explain in detail the operators and the elements to be considered.

Structuring element

The structuring element is a matrix that identifies the pixel in the image being processed and defines the neighbourhood used in the processing of each pixel. The size and shape of the structuring element is chosen to correspond to the same size and shape as the objects that need to be processes in the input image. In the figure below there is illustrated an example of flat binary masks. However, when applying morphological transformations over greyscale images this SE can be non-flat, where, each neighbour has an associated height.

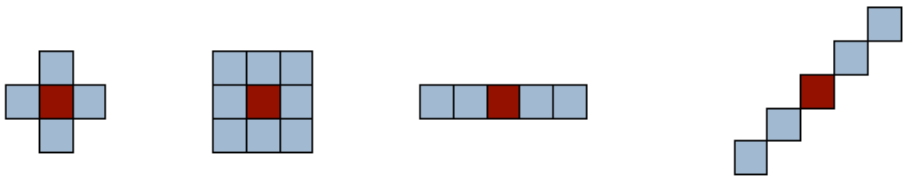

Figure 2.1. Example of symmetrical structuring elements. Source [10]

Erosion

The basic effect of this operator on a binary image is to erode away the boundaries of regions of foreground pixels. Thus, areas of foreground pixels shrink in size, and holes within those areas become larger.

If for every pixel in the structuring element, the corresponding pixel in the image underneath is a foreground pixel, then the input pixel is left as it is. If any of the corresponding pixels in the image are background, however, the input pixel is also set to background value.

Grayscale erosion with a flat disk shaped structuring element will generally darken the image; bright regions surrounded by dark regions shrink in size, and dark regions surrounded by bright regions grow in size. Small bright spots in images will disappear as they are eroded away down to the surrounding intensity value, and small dark spots will become larger spots. The effect is most marked at places in the image where the intensity changes rapidly, and regions of fairly uniform intensity will be left more or less unchanged except for at their edges.

Dilation

The main effect of this operator on binary images is to gradually enlarge the boundaries of regions of foreground pixels, meaning that areas of foreground pixels grow in size while holes or background pixels within those regions become smaller.

In binary dilation, for every position at which the structuring element is placed, if any of the image pixels comprised within the SE correspond to a foreground pixel, the input pixel will also be set to foreground; however if they are all background pixels, the input image will be left as it is

Grayscale dilation with a flat disk shaped structuring element generally brightens the image. Bright regions surrounded by dark regions grow in size, and dark regions surrounded by bright regions shrink. Small dark spots in images will disappear as they are `filled in' to the surrounding intensity value. Small bright spots will become larger spots.

Both erosion and dilation can be considered as dual operators, since dilating the foreground pixels is equal to eroding the background pixels.

Opening

While erosion can be used to eliminate small clumps of undesirable foreground pixels, e.g. `salt noise', quite effectively, it has the big disadvantage that it will affect all regions of foreground pixels indiscriminately. Opening gets around this by performing both an erosion and a dilation on the image. The effect of opening can be quite easily visualized. Imagine taking the structuring element and sliding it around inside each foreground region, without changing its orientation. All pixels which can be covered by the structuring element with the structuring element being entirely within the foreground region will be preserved. However, all foreground pixels which cannot be reached by the structuring element without parts of it moving out of the foreground region will be eroded away. After the opening has been carried out, the new boundaries of foreground regions will all be such that the structuring element fits inside them, and so further openings with the same element have no effect. The property is known as idempotence.

As we have seen, opening can be very useful for separating out particularly shaped objects from the background, but it is far from being a universal 2-D object recognizer/segmenter. For instance, if we try and use a long thin structuring element to locate, say, pencils in our image, any one such element will only find pencils at a particular orientation. If it is necessary to find pencils at other orientations, then differently oriented elements must be used to look for each desired orientation. It is also necessary to be very careful that the structuring element chosen does not eliminate too many desirable objects, or retain too many undesirable ones, and sometimes this can be a delicate or even impossible balance.

Closing

One of the uses of dilation is to fill in small background colour holes in images, e.g. `pepper noise'. One of the problems with doing this, however, is that the dilation will also distort all regions of pixels indiscriminately. By performing an erosion on the image after the dilation, i.e. a closing, we reduce some of this effect. The effect of closing can be quite easily visualized. Imagine taking the structuring element and sliding it around outside each foreground region, without changing its orientation. For any background boundary point, if the structuring element can be made to touch that point, without any part of the element

being inside a foreground region, then that point remains background. If this is not possible, then the pixel is set to foreground. After the closing has been carried out the background region will be such that the structuring element can be made to cover any point in the background without any part of it also covering a foreground point, and so further closings will have no effect. The closing operator is also idempotent.

Closing can sometimes be used to selectively fill in particular background regions of an image. Whether or not this can be done depends upon whether a suitable structuring element can be found that fits well inside regions that are to be preserved, but doesn't fit inside regions that are to be removed.

In **Figure 2.2**, a 1-Dimensional example, in analogy to what happens in 2D and 3D with image pixel values, there is presented a comparison between these four previous morphological operators. The original sample is presented in a black dotted line and the dilation, erosion, opening and closing are accordingly expressed in a red dotted line, a blue dotted line, a red line and a blue line.

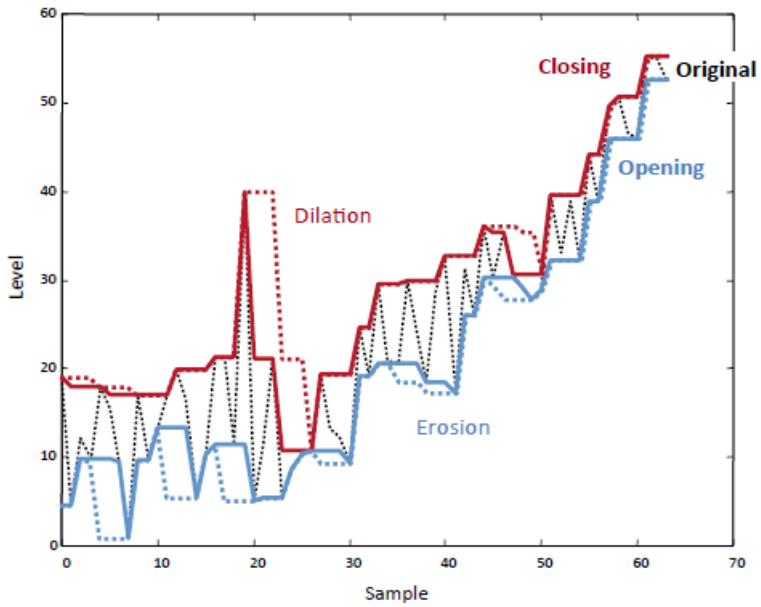

Figure 2.2. Comparison between erosion, dilation, opening and closing. Source: [10]

2.2. Deep Learning: Convolutional neural networks

2.2.1. Background

The origins of Convolutional Neural Networks can date back to 1998 with the publication of a paper written by Yann LeCun, León Bottou, Yoshua Bengio and Patrick Haffner [4], who built a neural network designed for recognising digits, but it has not been until recently that they have gained a great importance in computer vision. The key innovations to as why this has been this way is: computation time and available datasets.

Back in 1999, the transistor count [11] for potent computers of the time was of the order of 10^6 whereas now they reach an order of 10^9, meaning faster and faster computers.

On the other hand, for developing and training CNNs large amount of data and labels are needed. In 2009 researchers from the Computer Science department at Princeton University attending the Conference on Computer Vision and Pattern Recognition (CVPR), released the largest image database up until that moment: ImageNet. Currently it contains around 22 thousand object categories with over 14 million images. From that moment on, ImageNet gave way for computers to become almost as good as humans in visual recognition tasks.

The breakthrough moment for Convolutional Neural Netwoks, as said before, was in 2012 when the AlexNet [12] won the ImageNet Large Scale Visual Recognition Challenge with a 7 layer CNN by a margin of 10.8 points.

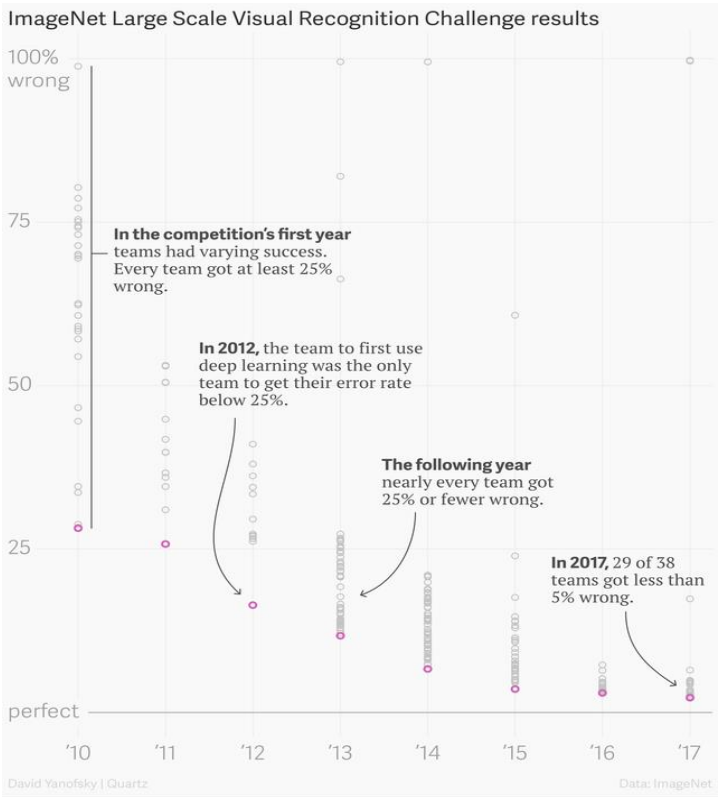

Figure 2.3. Error rate evolution in the ImageNet classification contest throughout the years**.** Source [13]

2.2.2. Convolutional Neural Networks

Convolutional Neural Networks (ConvNets or CNNs) are a category of Neural Networks, which in core, are systems inspired by the neuron connections in the human brain. They consist of a succession of different processes or layers operating in parallel, where the output of one layer is the input of the next (further explanation of the layers will be done in following sections).

The basic unit of computation in a CNN is the neuron, often called a node or unit. It receives input from some other nodes, or from an external source, and computes an output. Each input has an associated weight (w), which is assigned on the basis of its relative importance to other inputs. The node applies a function (transfer function) to the weighted sum of its inputs as shown in the next figure.

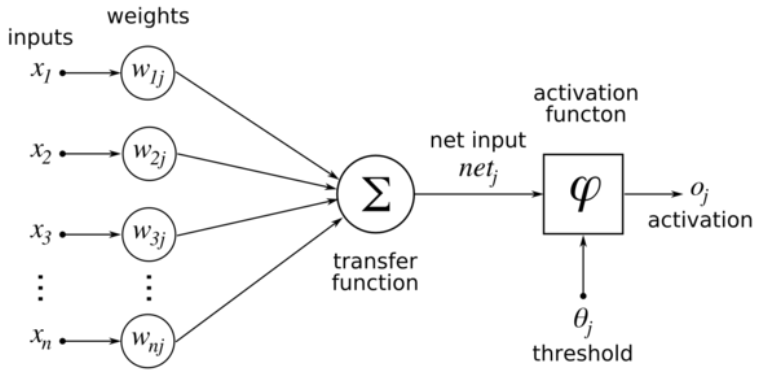

Figure 2.4. Single neuron scheme. Source: [14]

These networks have proven very effective in areas such as image recognition and classification. Specifically, ConvNets have been successful in identifying faces, objects and several Natural Language Processing tasks such as sentence classification, apart from powering vision in robots and self-driving cars. Contrary to previous methods, ConvNets learn directly from the image data that enters the system, therefore eliminating the need for manual feature extraction.

There are two main stages in ConvNets: test and train. During the first stage, a machine learning classifier ingests all the data, summarizes it and outputs a model containing the knowledge of how to recognize and classify these object categories. Then in the test phase, this trained model can be applied on new images to recognize all the objects.

The first important element in deep learning is a good and complete dataset. Visual data is very complex as there are high dimension of inputs, therefore if there is not a sufficient amount of data overfitting can happen very fast. This is a state where a model begins to "memorize" the training data rather than "learning" it, fitting the model too tightly and exclusively to the dataset images.

2.2.2.1. Architecture

As stated before, ConvNets consist of a series of layers where the three main types of layers are used are: the Convolutional Layer, the Pooling Layer, and the Fully-Connected Layer. ConvNets typically alternate between the first two types and are followed by the fully connected layer for classification.

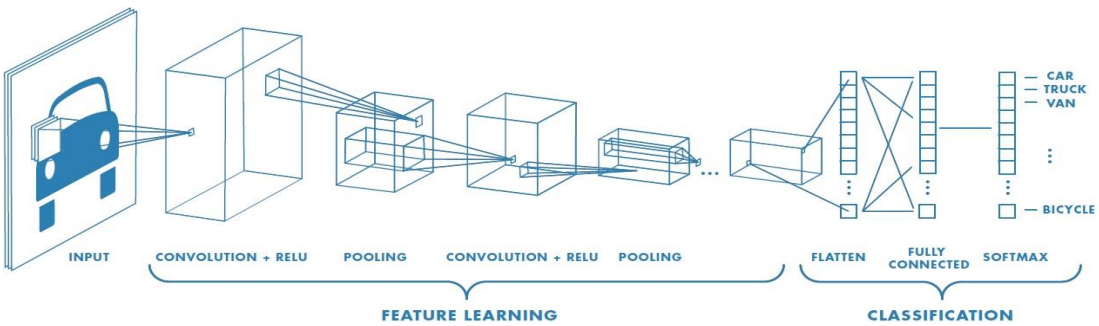

Figure 2.5. Diagram of a Convolutional Neural Network. Source [15]

Together, these layers extract the useful features from the images, introduce non-linearity in our network and reduce feature dimension while aiming to make the features somewhat equivariant to scale and translation.

2.2.2.1.1. Convolutional Layer

The first layer in CNNs is always a convolutional layer, which apply a specified number of convolution filters to the input image. The mechanism is as follows. The filter or kernel is a matrix of numbers of the same dimension as the input image. This filter then slides along every pixel value and multiplies each number of the filter with the corresponding pixel value it is placed on, computing element wise multiplications. At every pixel position, each multiplication is summed up to be a single element of the output matrix, called feature map, as illustrated in the figure below.

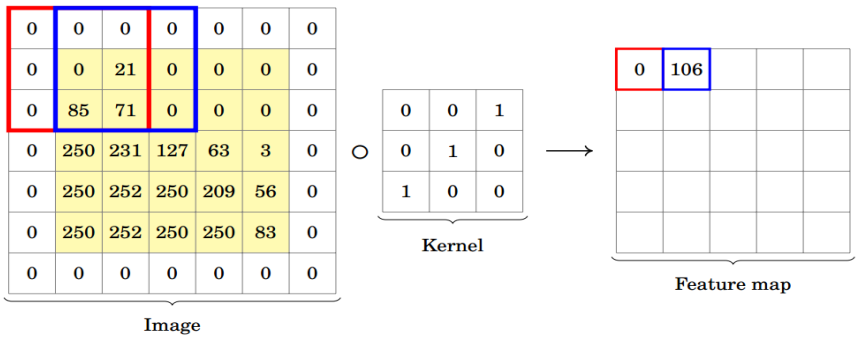

Figure 2.6. Practical example of a zero-padded convolutional layer. Source [16]

The input image is typically either 2-dimensional or 3-dimensional depending on whether it is a grayscale image or a colour image. The produced feature map, however, will always be 2-dimensional.

As said before, the filters are used to detect certain features in an image. Each layer can have as many filters as desired, so each feature map produced by convolving the input image with the filter will be stacked on top of the other. For example, if we have 12 different filters applied over an N x M x Z image in one layer, the output will be a N x M x 12 matrix (if zero padding is used as in the previous figure). By using more filters, special dimensions are preserved better.

There are many parameters that can be considered when building this layer, for example, what to do with the edges of the image when convolving with the filter. Typical options are: zero-padding, where an extra margin of zeros is added on the border of the image as seen in **Figure 2.6**, mirroring where the extra margin of pixels “mirror” the original pixels they are touching, (for example in **Figure 2.6**, the top pixels in the blue square would be [0, 21, 0]), or simply applying the convolution without adding extra pixels; in this case, the feature map will have 2 pixels less in each dimension.

Another parameter to take into account is the stride. This is the number of pixels the filter matrix moves each time we slide it to another position over the input image. Typically the stride is 1 because it has been proven to work better in practice. Additionally, as already mentioned stride 1 allows us to leave all spatial down-sampling to the POOL layers, with the convolutional layers only transforming the input volume depth-wise.

2.2.2.1.2. Activation Function

After every convolutional layer, an element wise activation function is applied. If the data we wish to model is non-linear then we need to account for that in our model, so by applying an activation function we break the linearity of the network allowing it to learn more complex functions than linear regression.

The three most typical activation functions are Sigmoid or Logistic, Tanhor Hyperbolic tangent and ReLu or Rectified Linear Units.

Sigmoid function: It is an activation function of the form $f(x) = \frac{1}{1+e^{-x}}$, limiting the output between 0 and 1. Between X values -2 to 2, Y values are very steep, which means that any small changes in the values of X in that region will cause values of Y to change significantly. However, towards either end of the sigmoid function, the Y values tend to respond very less to changes in X so the gradient in those regions is going to be small. This gives rise to a problem of "vanishing gradients", where network refuses to learn further or is drastically slow.

Hyperbolic tangent: Its mathematical formula is $f(x) = \frac{2}{1+e^{-2x}} - 1$, a scaled sigmoid function. Its output is zero-centered, ranging between -1 and 1. However, just like the sigmoid function tanh also has the "vanishing gradient" problem.

ReLU: This is the activation function mostly used in ConvNets. It function follows the formula: f(x) = max(0,x) so it takes the maximum value between the value itself and 0. As it involves simpler mathematical operations it is less expensive computationally then the previous two, and the problem of the "vanishing gradient" cannot happen.

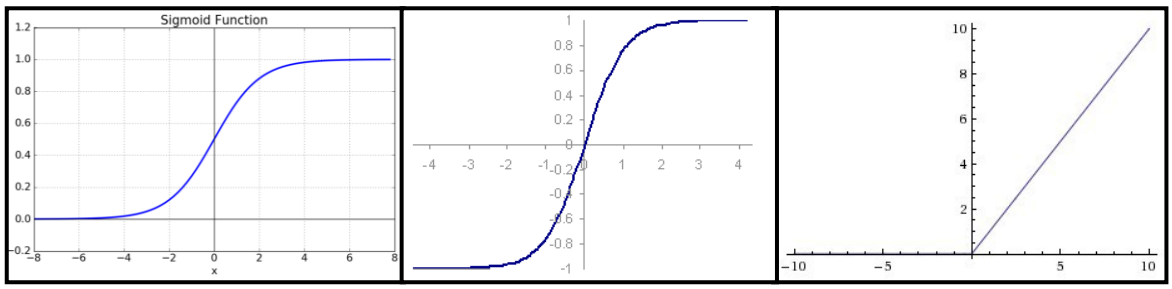

Figure 2.7. Sigmoid function on the left, hyperbolic tangent in the middle, ReLU on the right

2.2.2.1.3. Pooling layer

Spatial Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum etc.

In the case of Max Pooling, we define a spatial neighbourhood (for example, a 2×2 window) and take the largest element from the rectified feature map within that window. Instead of taking the largest element we could also take the average (Average Pooling) or sum of all elements in that window. In practice, Max Pooling has been shown to work better. Figure 2.5 shows an example of Max Pooling operation on a Rectified Feature map (obtained after convolution + ReLU operation) by using a 2×2 window and stride 2.

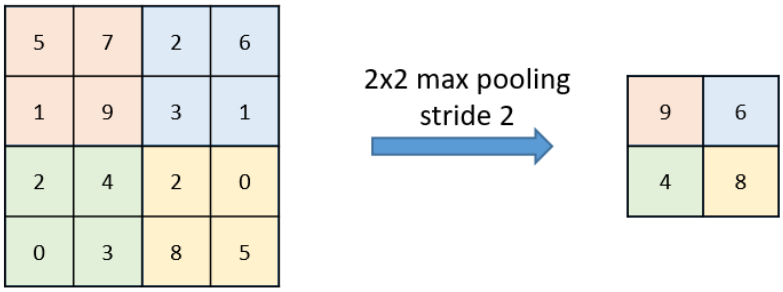

Figure 2.8. Max pooling. Source [17]

2.2.2.1.4. Fully connected layer

After several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. The term “Fully Connected” implies that every neuron in the previous layer is connected to every neuron on the next layer. The output from the convolutional and pooling layers represent high-level features of the input image. The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset.

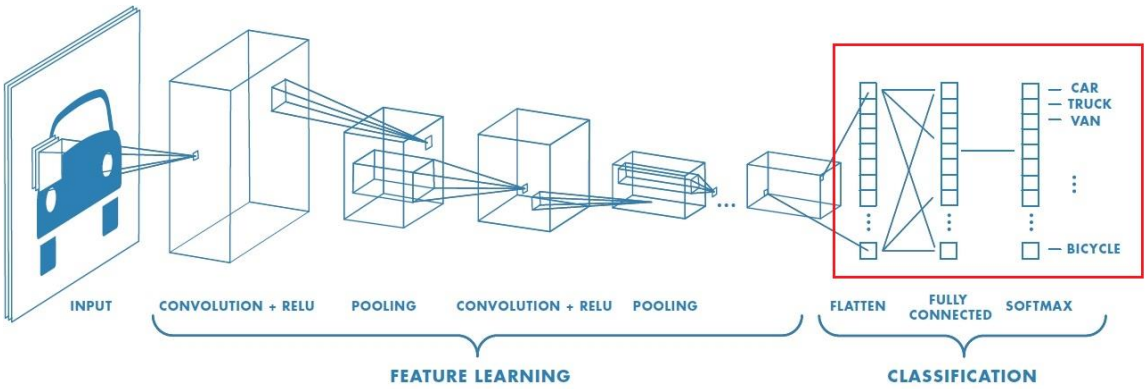

Figure 2.9. Diagram of a Convolutional Neural Network. Classification focus. Source [15]

2.2.3. Training using Backpropagation

Backward Propagation, often abbreviated as BackPropagation or BackProp is one of the several ways in which a neural network can be trained. In supervised learning, the training set is labelled meaning that for some given inputs, we know the desired/expected output (label).

Backpropagation can be separated into 4 distinctive sections: the forward pass, the loss function, the backward pass, and the weight update. Initially all the filters and weights are randomly assigned. For every input in the training dataset, the ConvNet is activated and finds the output probabilities for each class. Since weights are randomly assigned for the first training example, output probabilities are also random. Then, this output is compared with the desired output that we already know from the labels, and the total error, or the loss function, is “propagated” back to the previous layer (BackPropagation). In this step, the gradients of the error with respect to all weights in the network are calculated, and then

gradient descent optimization algorithms are used to update all filter values / weights and parameter values to minimize the output error. This process is repeated with all of the training images until the output error is below a predetermined threshold.

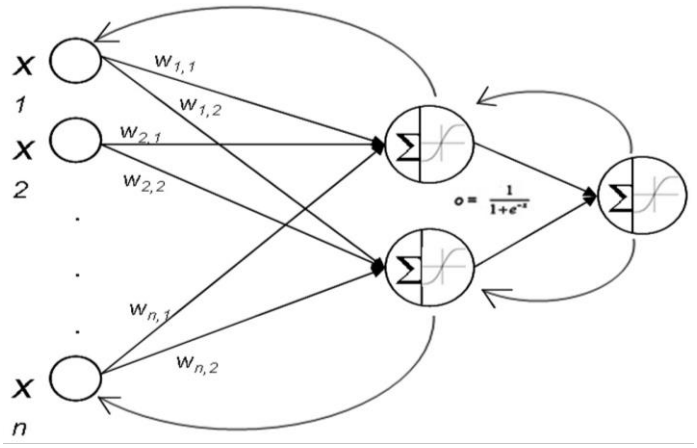

Figure 2.10. Backpropagtion diagram.

There are a few parameters to be considered in this section that can increase the performance of the training process. One of them are the optimizers, which are used to find the best coefficients that minimize the loss function. Stochastic gradient descent is typically used. Another parameter that can increase performance is the cost function, which is a measure that calculates how good the network did respect to the given input and the expected output. Choosing a good cost function can help to regulate the speed that the network learns (or the update step the weights take).

3. Methodology / project development:

In this section, we present the relevant methods used in the development of the project. We will first proceed to list the technologies and software tools used to set up our working environment. Then we will proceed to explain the methods and techniques used, as well as the steps taken to develop the software.

3.1. Technologies used

This section describes the software and machines used in the development of the project. In the first half, the project was developed on a laptop with the following specifications:

- **Operating system:** Windows 7 Ultimate
- **Processor:** Intel i7-2630QM 2GHz
- **RAM:** 8GB
- **Graphic card:** Radeon ™ HD 6770M

To develop the mathematical morphology and also to implement the labelling program [18] that Albert Aparicio created, we used Matlab 2016b programming engine with the following installed packages:

- Image processing toolbox v.9.5
- Statistics and Machine Learning Toolbox v.11

In the second half of the project, for implementing the deep learning techniques, the programming language that was used was Python 3.5, with tensorflow as the library to develop machine learning. The scripts were executed on the Image Processing Servers of the Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona (ETSETB). The packages needed for executing the code were the following:

- Numpy v.1.13.1
- Click v.6.7
- Pillow v.4.2.1
- Tensorflow-gpu v.1.2.1
- Scipy v.0.19.1
- Scikit-image v.0.13

3.2. Image Database

The images used for the development of this project have been provided by Berta Raventós Roca, a Biological Systems Engineering student from the Escola Superior d'Agricultura de Barcelona who performed a research in Leishmania parasite cultivation. To be able to observe and recognize the different cell components, an auxiliary technique used in microscopy for image contrast enhancement is applied. This technique is called staining. There are many staining methods available, though for this dataset of images Giemsa staining is used. The images have been captured using a mobile phone under a microscopic magnification of 50, and are stored in PNG format with a resolution that averages 1320px x 1900px. The total number of images in our dataset is of 45.

3.3. Basic image processing techniques

In order to create a piece of software able to detect Leishmaniasis parasites using more conventional methods that do not require powerful computers, the steps are the following: first of all, the input image will have to be processed in order to enhance it and remove any artifacts that might be present, such as noise, or to correct uneven illumination, allowing the most relevant objects to stand out. After this, we will need to locate and perform a segmentation of the parasites, if any, within the image to perform an individual examination of each detected object in order to evaluate whether the blood sample is infected with leishmaniasis or not.

Figure 3.1. Block diagram

The parasites and the cell's nucleus are the main focus for segmenting and obtaining the parasite density. When looking at Giemsa stained images it is easy to classify these two objects of interest as they can be observed in a darker shade than the cytoplasm and the background (**Figure 3.1**). However, this presents the first challenge since they both come in different shapes and sizes. On the one hand, the cell's nucleus is large and elliptical, typically easy to identify because they are isolated from each other; we might find touching cells but not touching nucleuses. On the other hand, the parasites can be presented in various forms as shown in the right side of **Figure 3.1**: when they are in their amastigote form, located within cells cytoplasm, they are typically small rounded dots but can often be mistaken as part of the non-homogeneous cytoplasm; when they are in their promastigote

form, outside of the cells, they can become elongated and in many occasions are grouped together, making it hard to separate from one another.

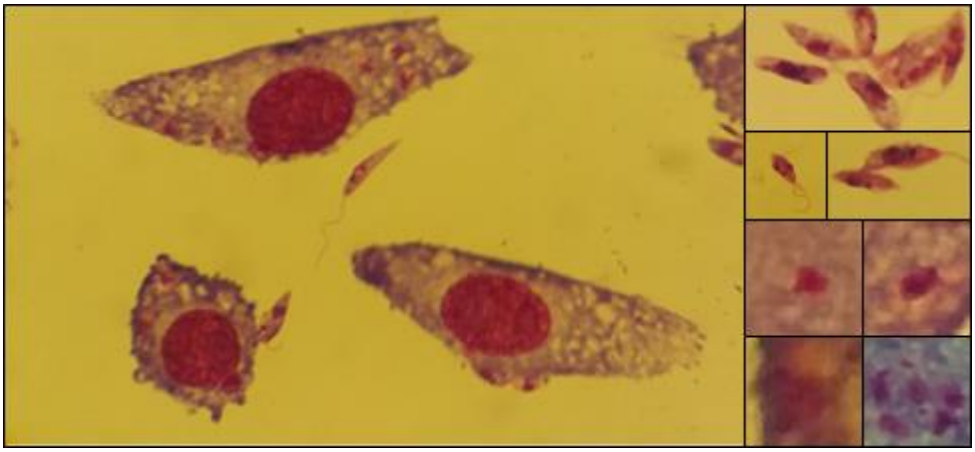

Figure 3.1. Giemsa stained image(left). Promastigotes (3 images top right). Amastigotes (4 images bottom right).

3.3.1. Image pre-processing

The main goal of the pre-processing is to maximise the useful information in an image while discarding the redundancies, in order to make the job of the feature extraction and classification blocks easier.

For this purpose, we have performed tests using two different approaches. At the end of this section we will describe which technique fits our needs the best.

The first proposed method aims to do cell segmentation to detect and separate the red blood cells and the promastigotes from the background. The second method takes advantage of the dominant colours in the image, to produce a binarized version containing the cells and the candidate parasites.

3.3.1.1. Watershed segmentation

There exist two basic ways of approaching image segmentation. The first one is boundary-based and detects local changes in the image, and the second one is region-based, that searches for pixel and region similarities.

The proposed method makes use of watershed segmentation consisting of “region growing”, as it has been used in many cases of cell segmentation. This transformation comes from the geological meaning of watershed, which is defined as a divide that separates adjacent catchment basins.

The basic idea of the watershed transformation is to imagine the image to segment as a topographic surface, where the grey-levels can be thought of as the height of the surface. If we make the analogy to the geological meaning, bright pixels are the lines that run along the top of the ridges while darker pixels are the catchment basins. The objective is to place a water source in each catchment basin and to start flooding until the different water sources meet, this way creating region barriers. There is a graphical example in **Figure 3.2** to understand the flooding concept, as it can be somewhat difficult to picture the brightness of an image as a topographical surface.

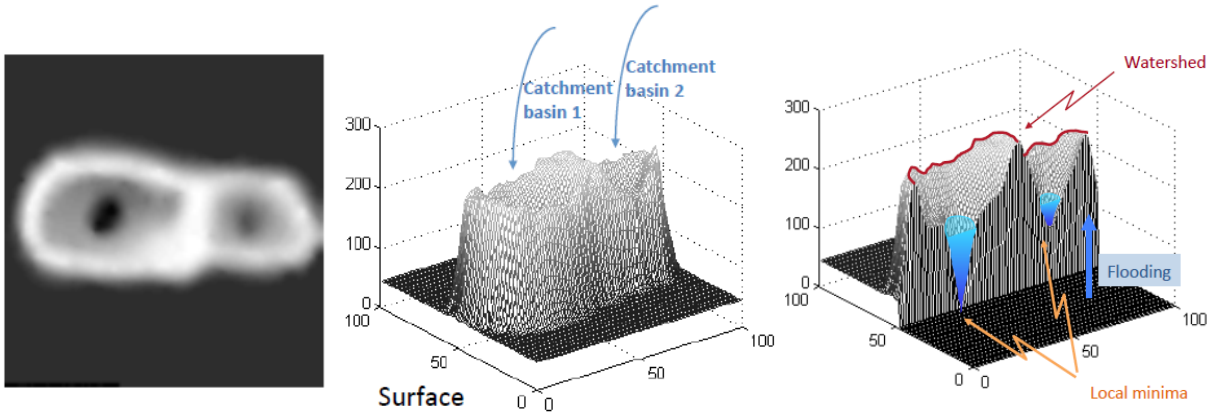

Figure 3.2. Original image (left). Topographic surface of the image (centre). Transversal cut of the topographic surface with water flooding (right). Source [26]

A typical phenomenon in watershed transformation is oversegmentation, where the image can become overly fragmented into subcomponents by creating many insignificant boundaries. Median filter is said to remove the impulsive noise from the image and therefore, prevent oversegmentation. The advantage of the median filter over others is that it preserves the borders of the segments making it beneficial for a proper shape extraction. So this is the first step to apply.

The use of the single watershed algorithm alone can also lead to oversegmentation, given that the input image is typically not as simple as the one in **Figure 3.2**. For overcoming this problem, marker-controlled segmentation can be used.

To find the markers, we transform the RGB image into a grayscale image, and produce a binarization using Otsu's method [??]. The resulting image will contain the objects to segment in white, and the background in black. This approach is based on the idea that machine vision systems often roughly "know" from other sources the location of the objects to be segmented, in this case the source is the colour.

After this, an image opening and a hole filling are also applied to the binarized image, as there are often regions within the cell area that have been incorrectly binarized as part of the background. We obtain the perimeter of the foreground object using an inbuilt method in Matlab to act as a ridge, and then dilate the binarized image to use as the catchment basin. **Figure 3.3.** Watershed process illustrates this process: a) shows the binarized image, b) shows the perimeter that has been extracted from the previous image, c) shows the superposition of the marker image and the perimeter on the original image and d) shows the final watershed segmented image.

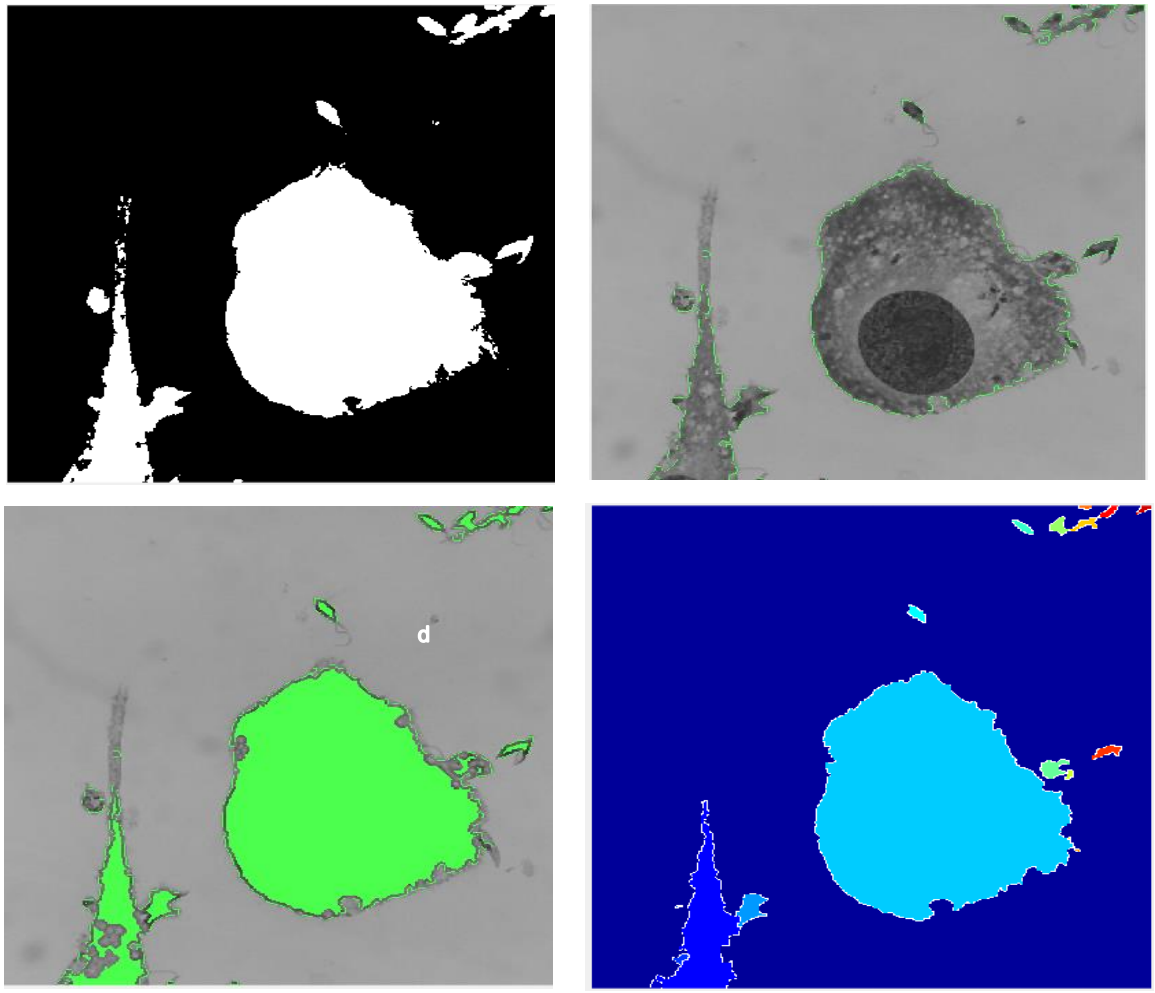

Figure 3.3. Watershed process

Note that this is useful to segment the promastigotes and the cells. If we observe the right top corner of the segmented image, we will see that it has done a pretty good job at segmenting the group of promastigotes. However, it does not take the amastigotes into account.

3.3.1.2. Colour segmentation

The second method, as mentioned before, takes advantage of the fact that the nucleuses and the parasites are tinted in the same colour tone while and the cell's cytoplasm and background are tinted in different tones.

We will be comparing two methods for image colour segmentation for extracting the "cell-clumps". The first one uses k-means algorithms to calculate the predominant colours in the image and assign every pixel to one of these colours. The second one uses histogram thresholding on different image colour components. The final step is to perform a binarization of the image.

K-means

This algorithm is widely used to solve classification problems. The main idea is to treat each object as having a location in space, in order to classify a given data set through a certain number of clusters, defined by k centroids. K-means finds partitions such that

objects within each cluster are as close to each other as possible, and as far from objects in other clusters as possible.

A number of fixed clusters are randomly placed over the data. After this, each pixel in the image is assigned to one of the clusters depending on its Euclidean distance to the centroid. After having assigned every pixel, the centroids are updated to fit according to the data assigned to that cluster. This process is iterated until the k centroids do not change their location any more or until maximum number of iterations is reached.

Figure 3.4. K-means iterations to find the centroids, from left to right & top to bottom. Source [10]

In **Figure 3.4** we can see an example of the centroids and data moving to create three separate classes. In our algorithm, depending on the number of clusters we have defined, the algorithm finds the k most likely colours given the set of pixels, and assigns a colour to the set of pixels in the same cluster. A maximum number of iterations set is of 150.

Once we have obtained the colour k-means segmented image, the next step is to binarize it in order to produce a mask for post-filtering. The built-in function in Matlab that performs the binarization follows Otsu's method so as to reduce the interclass variance. The output after each step in this method is shown in **Figure 3.5**. On the right is the original image, in the middle is the k-means colour segmented image, and on the right, is the image after applying a binarization.

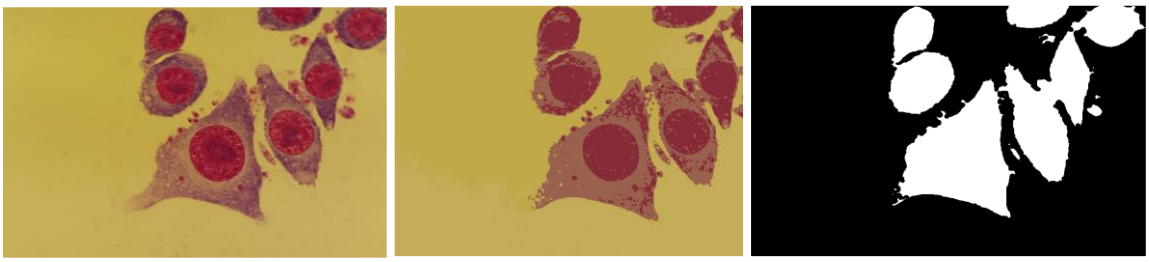

Figure 3.5. Colour k means algorithm, output images

Histogram thresholding

The second method takes advantage of the colour space. It tries to obtain the cells and parasites performing a binarization using a greyscale image containing the most relevant information possible.

We experimented with several colour spaces: RGB, HSL, HSV and YCbCr, and several transformations: we studied individual channels versus previously applying image contrast; we applied transformations to enhance individual colour components; we studied the

histograms of the colour components; we also tested colour component subtractions in the different colour spaces.

Finally we found that the best method was to subtract the green component from the red components in the RGB colour space. The image we obtained was a grayscale image where the parasites and the nucleuses where noted in bright pixels, the cells cytoplasm were painted in dark grey and the background of the images were practically all black. The three different regions were clearly separable by simply looking at the histograms of the images, since there were three distinctive peaks corresponding to each of the grey-levels mentioned before.

In observation of the generated R-G component images, we noted that many of the cells' cytoplasm contained holes showing through to the background. To solve this problem, given that the holes are a darker colour than the cytoplasm, a 'hole filling' was applied, where dark regions that are enclosed by lighter pixels take on the lighter pixels' value.

Finally, we proceed to binarize the image in the same way as before but using two thresholds instead of one. This way can segment the cells while also detecting the parasite candidates. To find the thresholds, Matlab has a built in function called *multithresh()* that implements Otsu's method to find the optimal threshold. In the figure below, the stages of this procedure can be observed. In the image on the left we have the image created from applying R-G. In the centre, the three notable peaks can be seen in the histogram. The last image is the segmentation produced after applying a threshold to the input image.

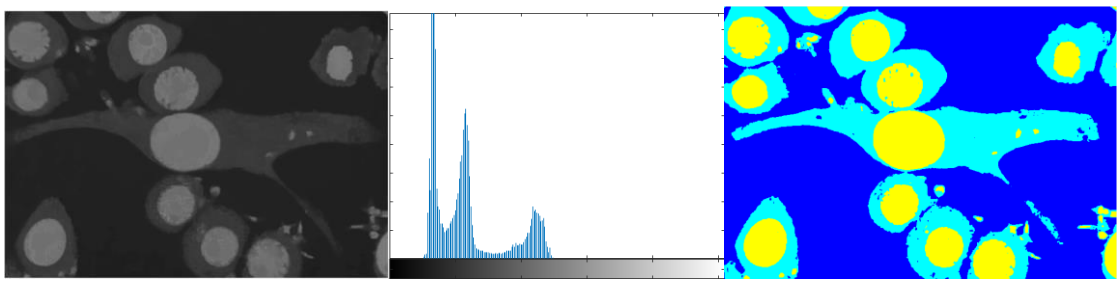

Figure 3.6. Red minus green component (left), histogram of the colour component difference (centre), segmentation using Otsu's method (right).

Considering the benefits and disadvantages of each of the three methods, we finally decide to use the histogram thresholding method over the watershed segmentation and the k-means colour segmentation. The main reason for this choice are the quality and efficiency of the results; the cytoplasm segmentation is not perfect, but it is sufficient for the context in which we need it, and moreover, all of the nucleuses and parasites can be localised, avoiding having to process the image a second time over in order to find these elements.

3.3.2. Finding candidate parasites

The next step in the process is to identify any possible candidates to be a parasite. As mentioned in the previous paragraph, through analysis of the histogram of the difference image we found that by using the same thresholding method as to detect the cytoplasm of the cells, the parasites could also be detected.

First, we created a mask where the nucleuses and the parasites were in white. At this point we want to discard the nucleuses, because the only objects of interest now are the parasites. In order to perform this task, we study the sizes of the particles in the image.

The technique is called Granulometry, and is achieved in a similar manner to that of which geologists use to determine the average particle size within soil, where the soil is sifted through a succession of filters of increasing size and the remains are collected after each pass.

Morphological granulometry, therefore, is achieved through a succession of openings with a structuring element of increasing size, and counting the remaining object surface area after each opening. The radius of the structuring element is increased until the remaining image has no more objects to eliminate. Below in **Figure 3.7** we can see some quantitative results from applying granulometry to the binary image. Image a) is the input image. In image b) we can see there are 7 distinctive peaks, which in turn correspond to the seven nucleuses of image a). Iis a simple plot of the particle pixel count in the vertical axis. The horixontal axis represents the amount of connected components in image, but is not ordered in any maner. Image c) represents the decay in white pixel count (y) against the structuring element radius (x). As it can be seen, there is a strong slope that starts at around a radius of 25; this is the point where the nucleuses start to dissappear. Image d) represents the gradient of the decay.

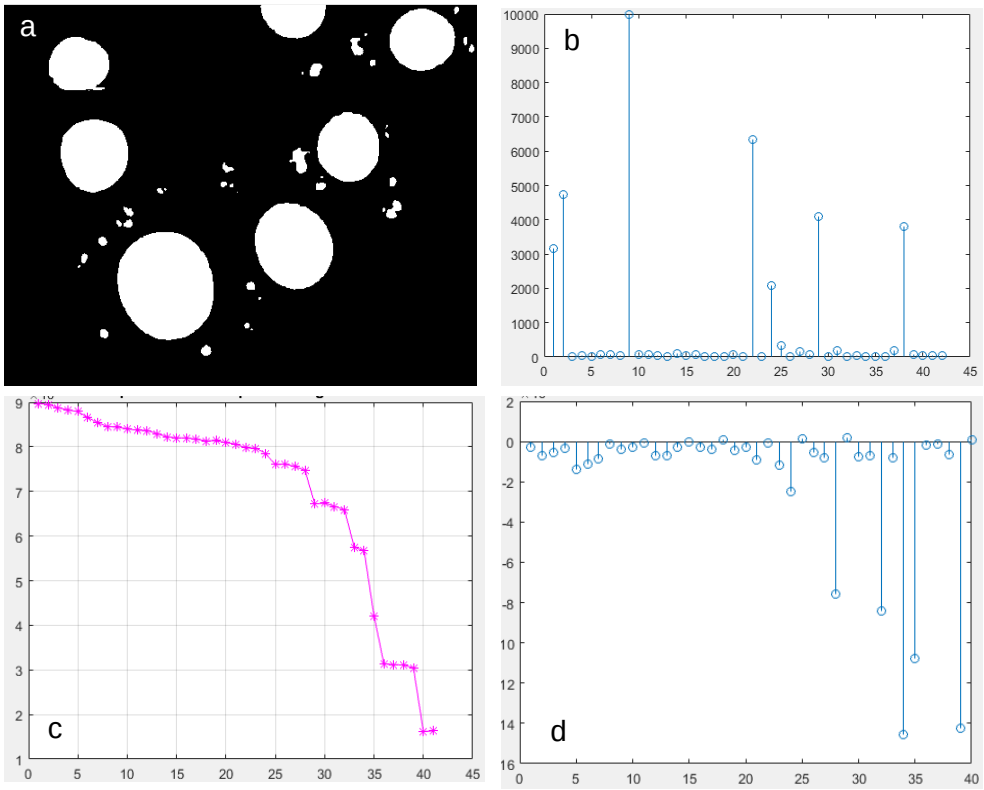

Figure 3.7. Particle size distribution

The structuring element we have chosen to use is a circumference, given that the nucleuses of the cells and of the parasites tend to have an elliptical geometry. Once the average particle size distribution is calculated, we remove the nucleuses in the image by eliminating the objects larger than the found radius value.

After this, we divide the candidate parasites into promastigotes and amastigotes, because the analysis for both of these forms of parasites will differ. To do this, we recover the binary mask containing the cytoplasm that we obtained in section 3.3.1.2, and we superimpose it with the binary parasite mask. The pixels that fall within large regions of cytoplasm are amastigotes, and the remaining parasites are classified as promastigotes. The objects that fall in the border of the image are eliminated.

A small note, is that the promastigotes that found themselves adhered to the exterior of the cytoplasm or located very close to it, were sometimes classified as amastigotes due to the fact that the region they fell upon on were large regions of wrongly classified cytoplasm.

One we have the two parasite classes, the next step is to extract small sub-images containing each of the candidates for individual examination. We followed the method proposed by Jaume Fernandez in his master thesis [5] for malaria parasite segmentation, where the sub-images are created around the centroids of each object in the binary mask. However, he also states the fact that the stained objects usually create more than one area. In other words, a single parasite may create several blotches that if treated independently, would lead to a detection of one too many parasites. In the case of leishmania parasites, this can also happen when the parasites are very close together.

To solve this, centroids that are very close together are treated as one, thus, the created sub-image will vary in size according to its content.

3.3.3. Individual analysis

What mainly distinguishes the parasites from the rest of the image is their colour.

We make the assumption that amastigotes and promastigotes will be darker in than the rest of the image, and this is generally true because of the staining technique used to obtain the images. We perform a closing of the image with a very small structuring element so as not to erase the parasite, to reduce at minimum any possible artefacts that may be present. Immediately after, a k-means colour classification (the same as in section 3.3.1.2) is applied, resulting in images similar to that in **Figure 3.8**. As mentioned before, the algorithm applied to detect amastigotes and promastigotes will be different. For instance, the number of clusters when applying the k-means classification will be of 3 and not 4 in the case of the promastigotes. If we closely inspect the middle images in **Figure 3.8**, the region of interest is also slightly different in each case. In the amastigote image, we just need to segment the darker area, whereas in the promastigote image, the area to segment is whatever is not background. For this reason, the binarization threshold is also different.

After applying the binarization with the corresponding threshold, a series of morphological operations are applied in this particular order: hole filling, elimination of small objects and dilation. The final segmentation can be observed in the right images of **Figure 3.8**.

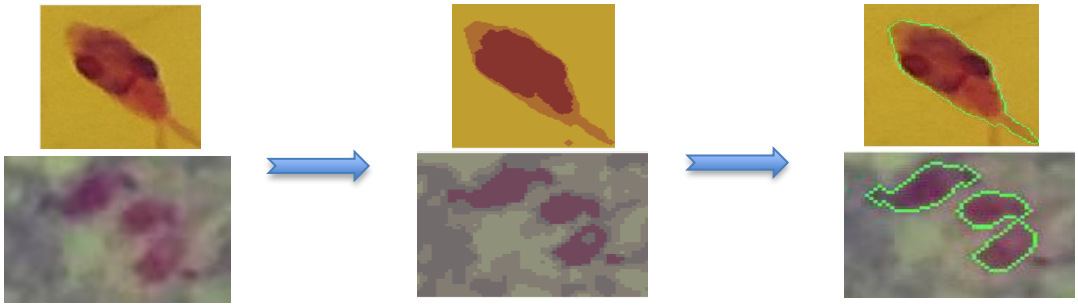

Figure 3.8. Amastigote and promastigote images after applying k-means colour classification and segmentation. From left to right: original images, k-means segmented image, detected parasites.

3.4. Deep learning techniques: U-net

Convolutional neural networks are typically used for classification tasks, mainly for assigning labels to whole images, and using thousands of images for training the networks. But in biomedical image processing it is most likely to need to assign a label to each pixel (pixel classification) allowing a localization of different regions in the image, while using few training images given that these are often a limited resource. Fully convolutional networks are often used, that is the fully connected layers are withdrawn.

U-net [19] presents an alternative when the set of available images is relatively small. In the original paper they stated that there were only about 30 annotated images available, and where this presents a main challenge for most Convolutional Neural Networks it was no setback for the U-net, as they relied on the use of data augmentation for a more efficient use of the annotated samples. This network was chosen precisely because the number of images in their dataset resembled the amount we had in ours: 45.

The basic U-net classifier that we use was downloaded from [20] and was initially developed and used for Radio Frequency Interference mitigation using deep convolutional neural networks [21]. Further improvements described in this project concerning the U-net have been developed in its integrity by Albert Aparicio.

In the following sections we will first explain the method used to label the images, because as we already know, this is a crucial part in deep learning, and then we will proceed to talk about the architecture and parameters of the U-net.

3.4.1. Image labelling

Deep learning for computer vision requires of a ground truth in order to train the model of the network. For this, Albert Aparicio, a research intern in the image processing department of the ETSETB, developed a labelling program [18] in Matlab which allows to manually draw different class regions of interest on the image, mapping each drawn object to a label. In our case, we have 8 different labels (**Figure 3.9**): background, cell nucleus, promastigote, amastigote, adhered parasites, cell cytoplasm, non-usable area and non-parasitic element. These last two labels are for when it is unclear what the object is (1), if the object is part of a parasite but is of no interest (2), or when there is only a fraction of the object in the image making it unidentifiable (3).

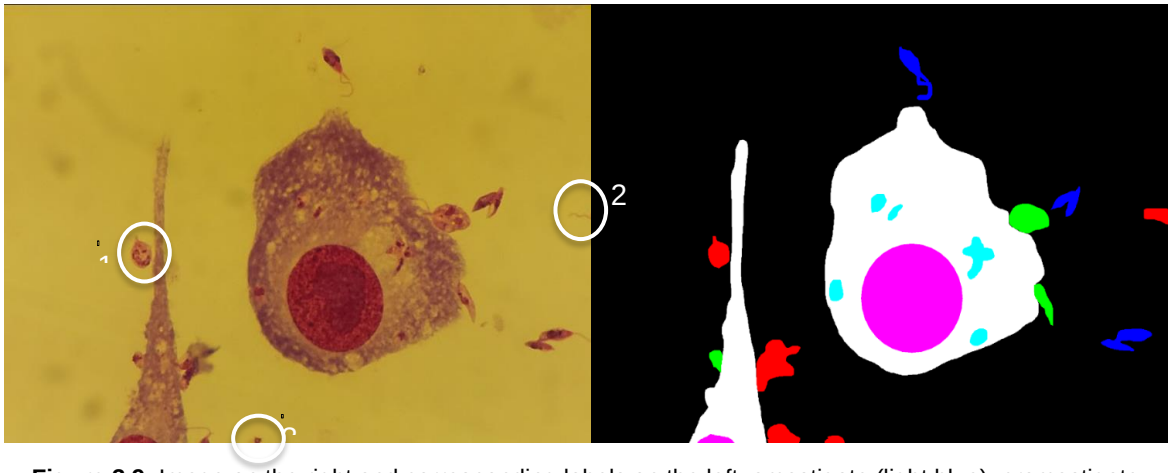

Figure 3.9. Image on the right and corresponding labels on the left: amastigote (light blue), promastigote (dark blue), nucleus (pink), cell cytoplasm (white), background (black), adhered parasites (green), non-usable area (red)

The 45 images in our dataset were split into two, and both Albert and I labelled half of them.

3.4.2. U-net structure

The architecture of the network consists of a large number of operations divided between the contracting (left) path and the expansion path (right), as seen in **Figure 3.10.** Each blue box corresponds to a multichannel feature map and the number of channels is denoted on top of these boxes.

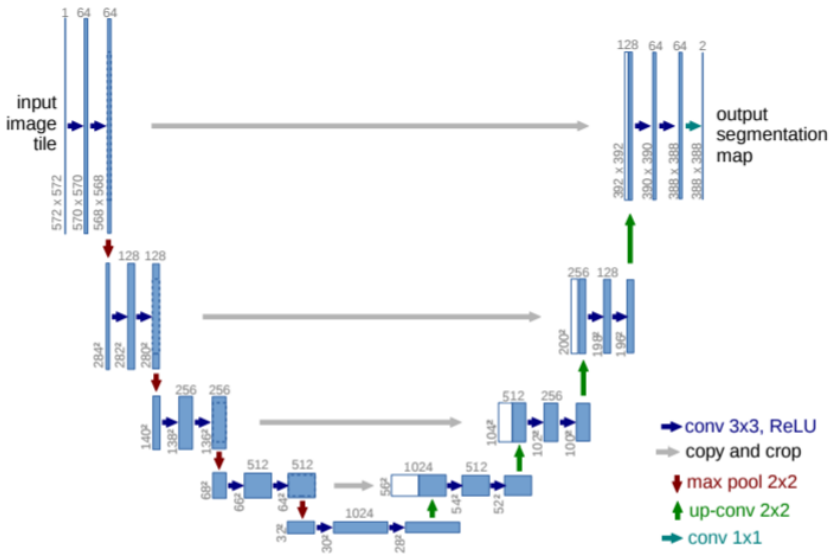

Figure 3.10. U-net architecture.

Most of the operations are standard 3x3 convolutions followed by a non-linear activation function. After the max-pooling layer, the number of feature channels is increased by a factor 2.

The more or less symmetrical u-shaped architecture of the network is due to the fact that high resolution features from the contracting path are combined with the upsampled output in the expansion path. As a large number of channels are used, it allows the network to propagate context information to higher resolution layers, giving the possibility to obtain

good pixel localisation and to produce a high-resolution segmentation map. A successive convolution layer can then learn to assemble a more precise output based on this information.

The original paper talks about performing random elastic deformations to the input images to produce data augmentation, since they were using a reduced dataset. In this project we decided to use mirroring and image rotation for simple computational reasons.

Given that our dataset contains large images of varying sizes, we also decided to introduce small fixed-sized patches into the system, and to discard patches that did not contain at least a 5% of parasite pixels. The reason for this second design choice to omit certain patches was that, initially, the model was only classifying pixels into background, cytoplasm and nucleus classes.

3.4.2.1. Parameters

In order to fine tune the model to perform the best it can, there are a few parameters that can be played around with. The most relevant ones being the following: the optimizer, the cost function, the number of layers, the number of initial filters, the patch size and the patch overlap, the number of epochs and training iterations, and variations or transformations within the input image.

In the remains of this section we will present some of the tested design options.

The goal of optimization, is to find the weights to minimize the loss function when doing backpropagation. In this project, we have used an optimizer called Adaptive Moment Estimation, or Adam [22], which is derived from the Root Mean Square Propagation (RMSProp) and the Adaptative Gradient (AdaGrad) optimizers. At this moment, it is one of the most used gradient descent optimizers, since It has been shown that Adam works well in practice and compares favourably to other adaptive learning-method algorithms, while reducing the performance time.

Various trainings have been launched using different parameters, but in general we have not noted any significant improvements. For instance, changing the number of epochs in a range from 10 to 50, modifying the number of training iterations starting at 50 to about 250, or variations in patch and overlap sizes from 100 to 250 and 5 to 30. The average patch size that was used the most, however, was of 150 because we measured the parasites and found that a region of these dimensions contained enough information to be able to extract the parasites.

A more recent test involved changing the input image colour space where the regions were more visible, or creating an image applying enhancement transformations on the individual colour spaces. However, as these are simple linear transformations to the original images it did not improve the results.

We also compared adding and removing the weights on the classes but without any luck.

The only slight changes noted have been when modifying the number of filters from 16 to 32 and incrementing the number of layers from 3 to 4. These results will be presented in the next section.

4. Results

To verify the accuracy of the software, there are a few previous measurement terms necessary for their understanding: precision and recall.

In pattern recognition, information retrieval and binary classification, precision is the fraction of relevant instances among the retrieved instances, while recall (also known as sensitivity) is the fraction of relevant instances that have been retrieved over the total number of relevant instances. Both precision and recall are therefore based on an understanding and measure of relevance. The last measure of accuracy is the F1 score, or F-score which is the harmonic average of the previous two measurements. The formulas of these values are the following:

$$Precision = \frac{TP}{TP+FP} \qquad Recall = \frac{TP}{TP+FN} \qquad F1 = 2 * \frac{Pecision*Recall}{Precision+Recall}$$

Where TP corresponds to true positives, which are the correctly detected parasites; FP are the false positives, which correspond to the objects that were classified as parasites when they were not really parasites; and FN are the false negatives, which correspond to the non-detected parasites that were really parasites.

4.1. Basic Image processing techniques

The data from the following tables has been obtained from the analysis of 10 images. The first two tables correspond to the precision and recall values of amastigotes and promastigotes:

Amastigote count	TP	FP	FN	Precision	Recall
91	80	51	11	**61.07%**	**87,9%**

Table 1. Precision and recall values for amastigotes

Promastigote count	TP	FP	FN	Precision	Recall
70	33	3	37	**91%**	**47,14%**

Table 2. Precision and recall values for promastigotes

Nevertheless, considering that all of the false negative promastigotes were in fact detected as amastigotes, we can reconsider the previous results and re-calculate these measurements according to the number of total parasites detected, regardless of their form:

Total parasite count	TP	FP	FN	Precision	Recall
161	150	17	11	**89.92%**	**93.16%**

Table 3. Precision and recall values for parasites

4.2. U-net

Next, we will present the results obtained from executing the U-net with different parameters. We have used 43 images for training the network and 2 images for test with their respective patches. The support column indicates the number of occurrences of each class in the ground truth.

We have chosen to display the two most relevant results:

- Batch size: 50
- Epochs: 20
- Patch size: 150
- Patch overlap: 10
- Layers: 3
- Initial filters: 16
- Optimizer: Adam
- Cost function: class-weights

Class Name	Precision	Recall	F1 score	Support
Background	0.5288	0.513494	0.47361	6284.41
Non-usable area	0	0	0	230.252
No-parasite	0	0	0	0
Cytoplasm	0.197406	0.406505	0.228658	2229.55
Nucleus	0.14978	0.13612	0.13803	1047.82
Promastigote	0	0	0	661.798
Adhered parasite	0	0	0	291.826
Amastigote	0	0	0	918.347

Table 4. First set of results from U-net

In the table below are the results of the U-net with the following parameters:

- Batch size: 60
- Epochs: 20
- Patch size: 150
- Patch overlap: 10
- Layers: 4
- Initial filters: 32
- Optimizer: Adam
- Cost function: cross-entropy

Class Name	Precision	Recall	F1 score	Support
Background	0.399246	0.350512	0.36	97.833
Non-usable area	0	0	0	5.733
No-parasite	0	0	0	0
Cytoplasm	0.2422	0.3224	0.250108	46.553
Nucleus	0.0978	0.081	0.0818	17.606
Promastigote	0.01496	0.010597	0.0104	7.922
Adhered parasite	0	0	0	7.814
Amastigote	0.0095	0.00883	0.0096	12.5333

Table 5. Second set of results from U-net

And in **Figure 4.1** you can observe the previous results represented as images. The image on the left corresponds to the original labels. The image in the middle corresponds to the **Table 4** results and the image on the left corresponds to the results in **Table 5**.

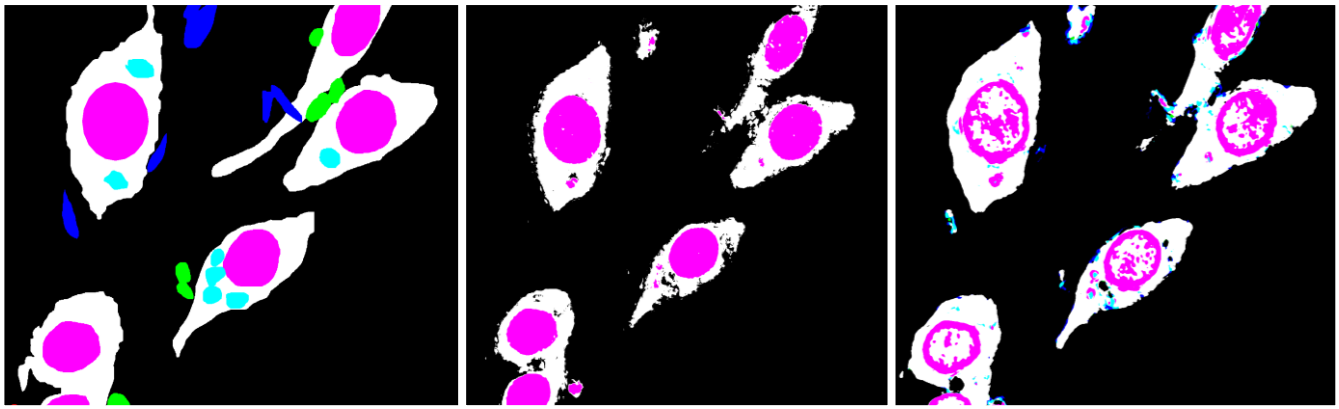

Figure 4.1. Image results from U-net classification

It is obvious that the best segmentation is the one from the picture in the middle using the parameters in the first table, since the nucleuses are detected almost to perfection, and the cytoplasm of the cells are not far off.

However, considering that many training iterations have been launched with no improvements in the results, we find that the results in the last image take us a step closer to obtaining the class segmentation that we are aiming for. The nucleuses are not perfect, but at least two more classes appear (amastigotes and promastigotes).

5. Budget

An important section in any kind of project being developed is the budget. For this reason, in our project we have implemented free software tools and educational versions of the software. On the one hand, Python and tensorflow are open source tools that can be downloaded by anyone through their websites [23] [24], and are available for a great variety of operating systems and versions. On the other hand, Matlab is a payed software, but it offers the possibility to purchase a student license for those taking courses in degree-granting institutions, therefore, the cost is less than expected.

First of all, we have to take into account all of the hours that have been put into this project. We have considered that the development of the system requires two engineers that are partially dedicated to this task, and a project coordinator that advises and supervises the progress of the project. For the engineers, supposing that they work work 20 hours a week for 30 weeks makes a total of 600 hours. For the project coordinator, supposing the amount of weekly hours spent is 5, which make a total of 150 hours.

Next we have to account for the material used. As we are using personal laptops, we will need to consider the amortization costs: we set the initial price of the laptop at 1304.6€ with a life expectancy of 5 years and a residual price of 175€. So the amortization value per year will be of 225.92, meaning that for 6 months the price will be of 112.96€.

A GPU and a server are also necessary in the development of the project for the training and testing of images in the deep learning part of the project. And finally, the Matlab license has to be added too.

	Cost	Total
Junior software engineers	600h x 12€/h	7200€
Project coordinator	150 x 20€/h	3000 €
Personal Laptops	112.96€ x 2	225.92€
GPU		200€
Server		400€
Student Matlab license	69€ x 2	138€
Light, internet and telephone	90€ x 6 months	540€
	Total:	11,703.92 €

Table 6. Budget analysis

6. Conclusions and future development:

The purpose of this project was to deepen our knowledge in image segmentation and classification techniques by implementing a parasite detection software. The two methods researched follow very different paths and include two very separated ways of approaching image segmentation, and as we have seen in our case, can also obtain very different results.

As we have seen with mathematical morphology, there is a wide set of available techniques that have been overly exploited during the years. The decisions made were based upon what fit the most according to our needs, but for sure could have been improved. For instance, in the pre-processing section we see that watershed segmentation obtains good results for segmenting the cells and the promastigotes, but does not segment the amastigotes. Perhaps with some more research we could have also achieved amastigote segmentation, and to some extent, improved our results.

To enhance the results from the U-net seems tricky at this point, but having read about the good results that it obtains in certain tasks, makes it hard to believe that it worked so poorly in our project. As there are no specific guidelines for resolving computer vision tasks, the solution to the incorrect detection issue may just be lying in a certain combination of different parameters to the ones used.

However, even though the final detection results were very inferior in the Convolutional Neural Network than with the basic image processing techniques, we would still bet on the novelty of deep learning, as it is an area in expansion and leaves the doors open to new ideas in the future.

6.1. Personal conclusion

This project was developed with the main objective of expanding my knowledge in image segmentation and classification techniques, and I can assure that this objective has been accomplished. I was first introduced to image processing techniques a few years ago, and was amazed by the unlimited spectrum of possibilities available at the tip of our fingers. Given that the world as we now know it revolves around computer vision, this was a good field to choose to do my final thesis on.

After developing this project, I can say that I have gained experience in project development and a lot of new information, especially regarding deep learning techniques as they were a novelty for me.

One of the main conclusions that I have been able to extract from this project, is the fact that a good initial planning is the key to success. Mine could have been improved, and also a bit more implication wouldn't have harmed. Overall, the sensation is a positive one. All experiences are positive.

Bibliography:

[1] W. H. Organization. [Online]. Available: http://www.who.int/mediacentre/factsheets/fs375/en/. [Último acceso: September 2017].

[2] WHO/Regional Office for South-East Asia, World Health Organisation, 2013. [Online]. Available: http://www.who.int/neglected_diseases/resources/B5042/en/. [Accessed: September 2017].

[3] H. A., « Pre-and post-treatment antibody levels in visceral leishmaniasis,» *Trans R Soc Trop Med Hyg.,* Sep-Oct 1990.

[4] De Almeida Silva L, Romero HD, Prata A, Costa RT, Nascimento E, Carvalho SF, Rodrigues V, «Immunologic tests in patients after clinical cure of visceral leishmaniasis,» *Am J Trop Med Hyg.,* October 2006.

[5] Jaume Fernàndez García, «Deep Learning Neural Networks in Malaria Diagnosis,» Barcelona, 2014.

[6] Berta Raventós Roca, «Desenvolupament de mètodes experimentals de cultius d'amastigots de Leishmania spp. per millorar la quantificació dels resultats experimentals.,» *Escola Superior d'Agricultura de Barcelona, UPC,* July 2016.

[7] Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (* = equal contribution) , «ImageNet Large Scale Visual Recognition Challenge,» *IJCV,* 2015.

[8] 2012. [Online]. Available: http://image-net.org/challenges/LSVRC/2012/results.html. [Accessed: September 2017].

[9] A. Haas, G. Matheron and J. Serra , «Morphologie Mathematique et granulometries en place,» *Annales de Mines,* 1967.

[10] Image and video Signal Processing (PIV) slides, Image Processing Group, TSC, ETSETB, UPC, 2014.

[11] Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Transistor_count. [Accessed: September 2017].

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, «ImageNet Classification with Deep Convolutional Networks,» 2012.

[13] D. Gershgorn, 2017. [Online]. Available: https://qz.com/1034972/the-data-that-changed-the-direction-of-ai-research-and-possibly-the-world/. [Accessed: September 2017].

[14] «Wikimedia Commons,» [Online]. Available: https://commons.wikimedia.org/wiki/File:ArtificialNeuronModel_english.png. [Accessed: September 2017].

[15] Mathworks. [Online]. Available: https://es.mathworks.com/discovery/convolutional-neural-network.html. [Accessed: October 2017].

[16] V. Pavlovsky, 2017. [Online]. Available: https://www.vaetas.cz/blog/intro-convolutional-neural-networks/. [Accessed: September 2017].

[17] Microsoft, «Python API for CNTK,» [Online]. Available: https://www.cntk.ai/pythondocs/CNTK_103D_MNIST_ConvolutionalNeuralNetwork.html. [Accessed: September 2017].

[18] A. Aparicio, 2017. [Online]. Available: https://upc-leishmaniosis.gitlab.io/image-labeler/. [Accessed: May 2017].

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, «U-Net: Convolutional Networks for Biomedical Image Segmentation,» 2015.

[20] J. Akeret, 2016. [Online]. Available: https://github.com/jakeret/tf_unet. [Accessed: July 2017].

[21] Joel Akeret, Chihway Chang, Aurelien Lucchi, Alexandre Refregier, «Radio frequency interference mitigation using deep convolutional neural networks,» *Published in Astronomy and Computing,* 2017.

[22] Zhang, S., Choromanska, A., & LeCun, Y., « Adam: a Method for Stochastic Optimization.,» *International Conference on Learning Representations,* 2015.

[23] [Online]. Available: https://www.python.org/downloads/. [Accessed: October 2017].

[24] [Online]. Available: https://www.tensorflow.org/. [Accessed: October 2017].

[25] J. L. Vázquez, H. L. Ayala and C. E. Schaerer, "Mathematical morphology for counting Trypanosoma cruzi amastigotes," in *39th Latin American Computing Conference (CLEI)*, Laboratorio de Computacion Científica y Aplicada, Facultad Politécnica, Universidad Nacional de Asunción, Campus UNA, San Lorenzo, Paraguay, 2013.

[26] Yann LeCun, León Bottou, Yoshua Bengio and Patrick Haffner, «Gradient-Based Learning Applied to Document Recognition,» *Proceedings of the IEEE,* November 1998.

[27] U. Karn, The data science blog, 2016. [Online]. Available: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/. [Accessed: October 2017].

[28] C. Spark, 2017. [Online]. Available: https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html. [Accessed: October 2017].

Glossary

- AdaGrad Adaptative Gradient
- Adam: Adaptive Moment Estimation
- CL: Cutaneous leishmaniasis
- CNN: Convolutional Neural Network
- ETSETB: Escola Tcnica Superior de Telecomunicacions de Barcelona
- FN: False negatives
- FP: False positives
- HSL: Hue, saturation and lightness colour space
- HSV: Hue, saturation and value colour space
- ILSVRC: ImageNet Large Scale Visual Recognition Challenge
- MM: Mathematical Morphology
- ReLu: Rectified Linear Unit
- RGB: Red, green and blue colour space
- RMSProp: Root Mean Square Propagation
- PNG: Portable network graphics
- SE: Structuring element
- TP: True positives
- VL: Visceral Leishmaniasis
- YCbCr: Luminance and chroma colour space