CHALMERS
UNIVERSITY OF TECHNOLOGY

Automated Detection of Immune Cells in IHC Images

Master's Thesis in Biomedical Engineering

Juan Pedro Vigueras-Guillén

Department of Signals & Systems
Chalmers University of Technology
Gothenburg, Sweden 2015
Technical report no: EX010/2015

REPORT NO. EX010/2015

Automated Detection of Immune Cells in IHC Images

Juan Pedro Vigueras-Guillén

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2015

Automated Detection of Immune Cells in IHC Images
Juan Pedro Vigueras-Guillén

Supervisor & Examiner: Olof Enqvist, Department of Signals and Systems

Technical report no: EX010/2015
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Typeset in LaTeX
Göteborg, Sweden 2015

Abstract

Immunohistochemistry (IHC) is the standard technique to study immune cells. IHC exploits the principle of antibodies binding specifically to antigens. This interaction is visualized by attaching markers, such as fluorescent dye or enzymes, to the antibodies

Several diseases related to the immune system can be studied using IHC. In this thesis, the work has been focused in an illness called Chronic Obstructive Pulmonary Disease (COPD). This is a progressive disease characterized by a persistent blockage of airflow from the lungs. COPD was the third cause of death in the world in 2012, and there is no effective treatment. Furthermore, the development of more effective treatments is hampered by the lack of knowledge of the disease, specially regarding how the damaged tissue reacts and becomes inflamed.

The aim of this project is to explore and implement an algorithm that detects cells in enzyme-IHC images of tonsil tissue from patients with COPD. Two different methods were suggested: the first one, which might be called ‘a rank-SVM detector’, only provides detection, whereas the second one, ‘a flooding detector’, also provides segmentation.

The first algorithm consists of a difference-of-Gaussians detector, a ranking support-vector-machine (rank-SVM) classifier, and a final selector that finds the best solution. The goal is to rank a set of points from the image, called hypotheses, so that those at the centre of cells obtain a higher rank and thus a threshold can be employed to perform the classification. An exhaustive search is done to obtain the best combination of features from those hypotheses, being the features under study the histogram of gradients, mean colour, standard deviation, and colour histogram.

The second algorithm, which is based on the previous one, includes a new block between the detector and the classifier to perform segmentation, which adapts the watershed method. A simple SVM is now implemented, and new colour and shape features are defined.

Both algorithms provide similar results, obtaining a sensitivity around 95% and a specificity around 99%. For the rank-SVM algorithm, the threshold can be changed, providing a higher sensitivity but a smaller specificity, or vice versa.

Keywords: Immunohistochemistry, COPD, Rank-SVM, Support Vector Machines, Difference of Gaussians, watersheds, Histogram of Gradients, classification.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Olof Enqvist for his support through the learning process of this master's thesis, for his patient, guidance, enthusiasm, motivation, and immense knowledge. I also would like to thank Jonas Erjefält and Caroline Sandén from Medetect AB for their help, providing the necessary data and making this thesis possible. Furthermore I would like to thank the personnel at MedTech West for providing a great working environment.

I also want to express my gratitude to the '*Fundación Margit y Folke Pehrzon*' for granting me a scholarship for the first year of this master's degree. Most probably I could not have afforded coming to Sweden without their help, and for that I am eternally grateful.

A special thanks to my parents, for their love and support. Many thanks to my brothers, friends, and colleagues.

Juan Pedro Vigueras-Guillén, Gothenburg, August 6, 2015

Contents

1 Introduction

1.1 Immunohistochemistry (IHC)

Immunohistochemistry (IHC) is an immuno-staining technique used to detect proteins (antigens) in tissue samples. IHC exploits the principle of antibodies binding specifically to antigens. This interaction is visualized by attaching markers, such as fluorescent dye, enzymes or colloidal gold, to the antibodies [1]. In other words, if a certain cell has certain antigens on its surface, this cell might be visualized if a particular antibody with some colour-reacting enzyme is applied to the cell. A microscope is used to obtain the images of the stained cells.

Fluorescent images are produced by using a fluorophore, such as fluorescein or rhodamine. A fluorophore is a chemical compound that can re-emit light upon light excitation. This technique was the first to be discovered in 1941 [2]. These images tend to be grey or colour (Figure 1.1-left). They possess a black background, the objects appear as bright spots, and usually have a very low signal to noise ratio (SNR). They are often used for in-vivo cell imaging, and several detection methods have been proposed in the past. A comparison between several unsupervised and supervised methods can be found in [3].

Other common methods in immunohistochemistry use enzymes, such as peroxidase (discovered in 1967 [4]) or alkaline phosphatase (discovered in 1978 [5]), which catalyse a colour-producing reaction. These images possess a white background, and the cells appear as brownish/bluish pixels depending on the enzyme used (Figure 1.1-right).

Depending on the specimen under investigation and the sensitivity required, different immunohistochemical methods can be used.

In this thesis work, an automated algorithm for cell detection for enzyme-IHC images will be developed. Knowing with certain precision the number of cells that react to the immunohistochemical process and their spatial distribution can be helpful for further research in several diseases related to the immune system. In this work, the images were

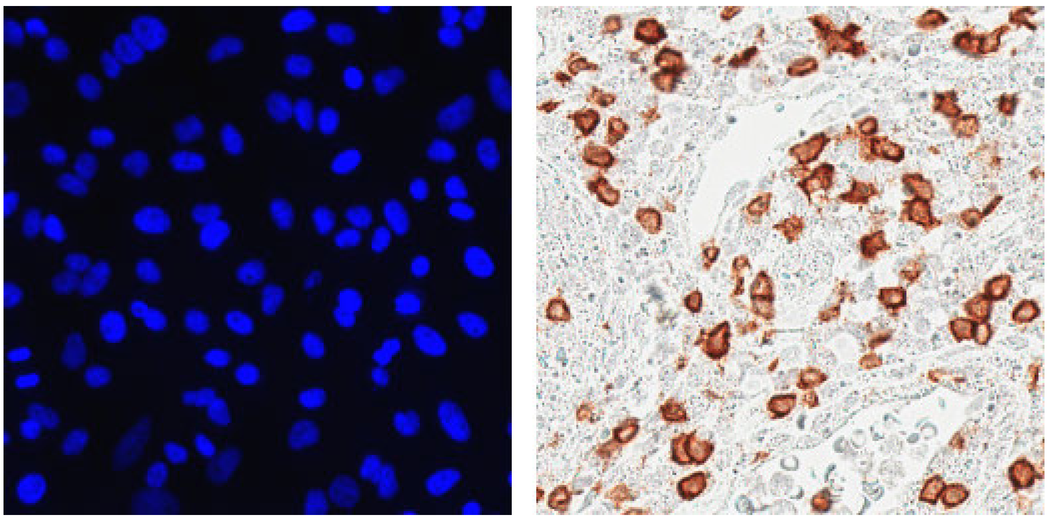


Figure 1.1: Immunohistochemical images stained with: **Left:** Fluorescent dye. **Right:** Enzymes

obtained from patients who suffer from 'chronic obstructive pulmonary disease' (COPD), although we believe our method could be applied to other diseases.

1.2 Chronic Obstructive Pulmonary Disease (COPD)

Chronic Obstructive Pulmonary Disease, also known as Chronic Obstructive Lung Disease (COLD), is a progressive disease characterized by a persistent blockage of airflow from the lungs. The main symptoms are coughing, production of large amount of mucus, shortness of breath and chest tightness [6]. It is believed that the main cause of this disease is cigarette smoking, although long-term exposure to other lung irritants, such as air pollution, chemical fumes, or dust, may also influence in its development. Genetics might also play a small role.

According to World Health Organization [7], COPD was the third cause of death in the world in 2012, killing around 3.1 million people per year (see Figure 1.2). This is specially significant in countries with lower-middle and upper-middle income level, with a ratio of 52 and 50 deaths per 100 000 inhabitants respectively. Moreover, total deaths from COPD are projected to increase by more than 30% in the next 10 years if no interventions are made [7]. COPD is barely present in countries with low income level due to the short life expectancy.

To understand how COPD works, it is necessary to know how the lungs operate in a healthy person. The air that goes down the trachea enters the lungs through two tubes called bronchi. Within the lungs, those bronchi subdivide to form secondary bronchi, and they continue to branch until ultimately giving rise to smaller bronchioles. Finally, those bronchioles subdivide into several alveolar ducts. The two or more alveoli that share a common opening to the alveolar duct are called alveolar sacs or air sacs. Inside

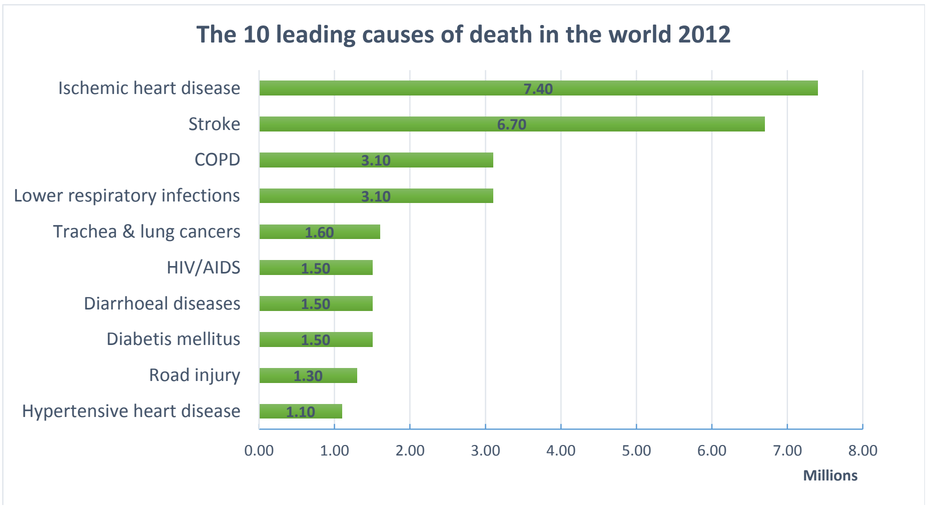


Figure 1.2: The top 10 leading causes of death in the world 2012. *Data retrieved from* [7]

the walls of those sacs there are small blood vessels called capillaries. When air reaches the alveolar sacs, oxygen passes through the sac walls into the blood in the capillaries. At the same time, carbon dioxide (a waste gas) moves from the capillaries into the sacs in a process called gas exchange. Those conducts and sacs are elastic, so when the person breathes in, the alveolar sacs are filled with air. In COPD the amount of air that goes through the lungs is significantly less than in a healthy person due to one or more of the following causes [6]:

- The walls between many of the air sacs are destroyed.
- The walls of the airways, as trachea and bronqui, become thick and inflamed, and lose their elastic quality.
- The airways make more mucus than usual, which can clog them.

The difference between a healthy lung and the lung of a COPD patient is shown in Figure 1.3.

The conditions of emphysema and chronic bronchitis are closely related to COPD. In emphysema, the walls between air sacs are damaged, which also destroys the walls of the air sacs. This generates bigger air sacs, and the gas exchange decreases. In chronic bronchitis, the airways are constantly irritated and inflamed, which generates thick mucus and makes breathing a hard process. Since most people who have COPD have both emphysema and chronic bronchitis, the term COPD is normally used to cover both conditions. There is no effective cure for COPD, although treatment can slow the progress.

Tobacco and other noxious particles cause lung inflammation, and a chronic inflammatory response will tend to induce tissue destruction over time, which will result in emphysema. It is believed that this chronic inflammation modifies the inflammatory

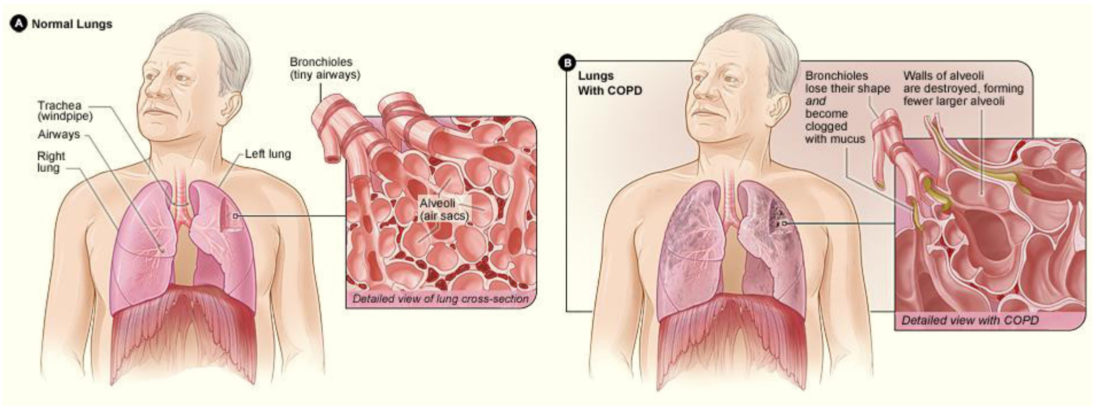

A Normal Lungs
Bronchioles (tiny airways)
Trachea (windpipe)
Airways
Right lung
Left lung
Alveoli (air sacs)
Detailed view of lung cross-section
B Lungs With COPD
Bronchioles lose their shape and become clogged with mucus
Walls of alveoli are destroyed, forming fewer larger alveoli
Detailed view with COPD

Figure 1.3: Healthy lungs and lungs with COPD. *Image retrieved from* [6]

response of the respiratory tract, although the mechanisms for this amplified response are not yet known completely [8].

Therefore, the lack of knowledge of how the disease develops, specially regarding how the damaged tissue reacts and becomes inflamed, is hampering the development of new treatments. The inflammatory cells involved are neutrophil granulocytes and macrophages, and Tc1 lymphocytes for those who smoke [8]. These cells release inflammatory mediators, which will attract inflammatory cells from the circulation, will amplify the inflammatory process, and will induce structural changes (the growth of air sacs). Therefore, the study of those immune responses can help to understand better this disease.

1.3 Aim

The aim of this project is to explore and implement an algorithm that detects cells in enzyme-IHC images of tonsil tissue from patients with COPD.

The efforts were focused in providing a good detection, leaving the segmentation of the cells as a secondary and optional goal that was achieved until certain point. Detection determines only the location of an object (the centre of the cells in this case), and segmentation entails the process of partitioning the image in set of pixels, also called segments or superpixels, that are meaningful (in this case, the set of pixels that forms a unique cell). This difference can be observed in Figure 1.4.

The microscope images used in this project were provided by the company Medetect AB at Medicon Village in Lund. An example of those images is shown in Figure 1.6.

In this company new technical innovations have made possible to identify different types of cells in the same tissue section. A microscope image of the unstained tissue is first captured. Then, stainings with different antibodies are added sequentially, and after each staining a new image is captured (see Figure 1.5). Observing those images it is clear that manual analysis is a time-consuming task. Therefore, an automated method is a necessary tool.

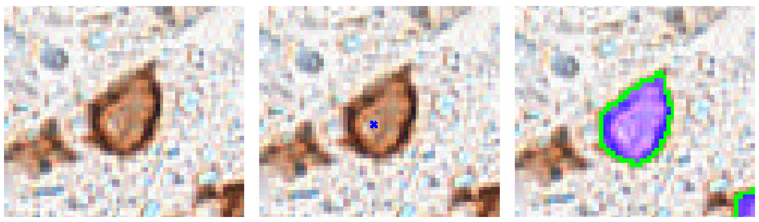


Figure 1.4: Left: Original image. **Centre**: Possible output for a cell detector algorithm. **Right**: Possible output for a cell segmentation algorithm.

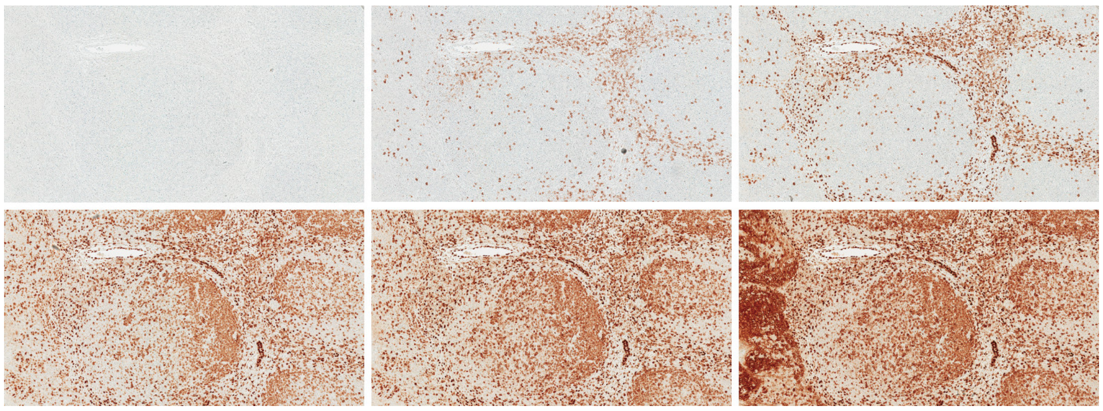


Figure 1.5: Unstained background (top left) and five sequential stainings using different biomarkers

1.4 MedTech West

This study was done at MedTech West (MTW), a joint venture founded in 2009 by Chalmers University of Technology, University of Gothenburg, University of Borås, Västra Götalandsregionen, and Sahlgrenska University Hospital. MTW aims to improve the quality and quantity of collaborative research projects in the area of medical technology. This study was done under the supervision of Assistant Prof. Olof Enqvist.

Algorithm development and testing was performed in the MATLAB programming environment.

1.5 Structure of the Thesis

The thesis is organized as follows:

- Chapter 2 deals with the theory behind both algorithms.
- Chapter 3 describes the structure of the algorithms.
- Chapter 4 addresses the design of the *raw detector*.

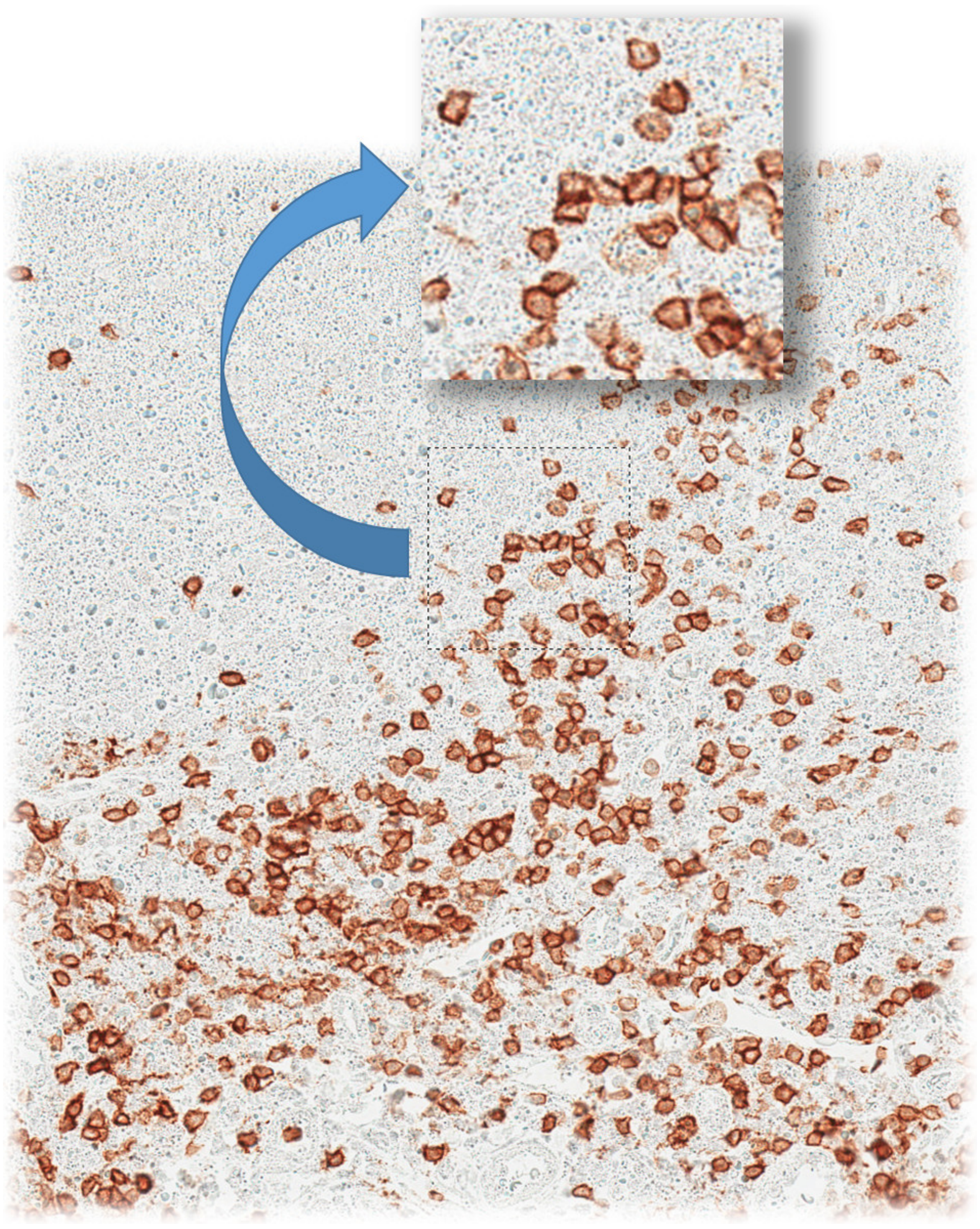


Figure 1.6: Example of the cell images to study

- Chapter 5 is involved with the methodology to extract and evaluate the features used in the *rank-SVM detector*.
- Chapter 6 deals with the segmentation algorithm in the *flooding detector*.
- Chapter 7 describes the features and the evaluation method used in the *flooding detector*.
- Chapter 8 addresses the last block of the algorithm, the *final selector*.
- Chapter 9 is concerned with the results from both approaches.
- Chapter 10 proposes future lines of work.

2

Theory

This chapter provides the theoretical background required to understand the rest of the thesis. In Section 2.1 the difference of Gaussians is described, which is employed in the *raw detector*. Section 2.2 introduces the watershed segmentation, which is the basis of our flooding algorithm. Some basic morphological operations will be used, so these are described in Section 2.3. The classifiers, SVM and rank-SVM, are described in Sections 2.4 and 2.5 respectively. Section 2.6 describes how the classifiers will be evaluated in this project. Finally, the maximum weighted independent set problem is addressed in Section 2.7, which is put into practice in the *final selector*.

2.1 Difference of Gaussians (DoG)

Difference of Gaussians (DoG) is commonly used as a grey-scale image enhancement algorithm [9]. This method subtracts one grey-scale image that has been blurred with a Gaussian kernel from the same image blurred with a different Gaussian. The blurred images are computed by convolving the original image with Gaussian filters with different standard deviations. Due to the distributive property of the convolution, DoG can also be obtained by subtracting one Gaussian from the other and then to convolve the result with the original image. A one-dimensional DoG response is shown in Figure 2.1. In images a 2D circular DoG is used.

The DoG algorithm can be expressed mathematically as follows: An image $I(x, y)$ is first smoothed by convolving it with a Gaussian kernel of certain standard deviation (σ_1),

$$G_{\sigma_1}(x, y) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{x^2 + y^2}{2\sigma_1^2}\right) , \tag{2.1}$$

to obtain

$$g_1(x, y) = G_{\sigma_1}(x, y) * I(x, y) . \tag{2.2}$$

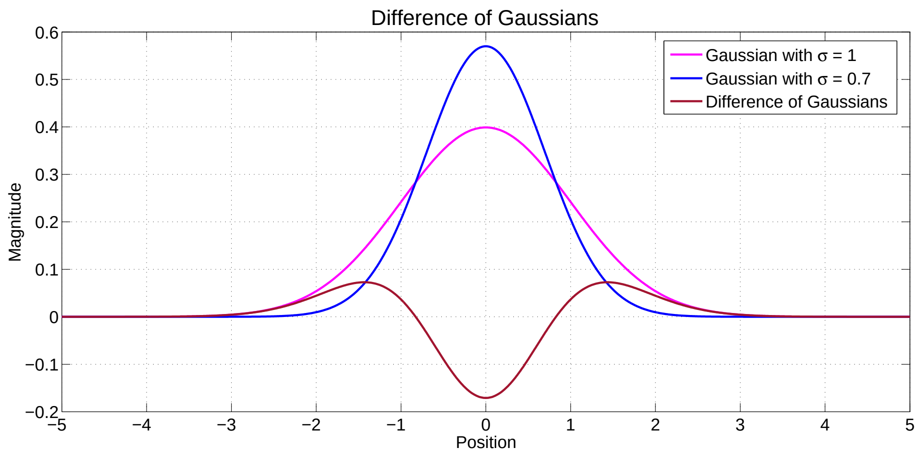


Figure 2.1: Difference of Gaussians (DoG) - Operator in one dimension

A second Gaussian kernel with a different standard deviation, $\sigma_2 < \sigma_1$, is also convolved with the original image

$$g_2(x,y) = G_{\sigma_2}(x,y) * I(x,y)\ . \tag{2.3}$$

Then, the Difference of Gaussians of the image $I(x,y)$, $\Gamma(x,y)$, is the difference between those two smoothed images

$$\Gamma(x,y) = g_1(x,y) - g_2(x,y) = G_{\sigma_1} * I(x,y) - G_{\sigma_2} * I(x,y) = (G_{\sigma_1} - G_{\sigma_2}) * I(x,y)\ ,$$

being the DoG operator:

$$DoG = G_{\sigma_1} - G_{\sigma_2} = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{x^2+y^2}{2\sigma_1^2}\right) - \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{x^2+y^2}{2\sigma_2^2}\right)\ . \tag{2.4}$$

Another application of the DoG algorithm is blob detection, which is the one used in this project.

2.1.1 Difference of Gaussians as a Blob Detector

A blob in image processing is a region of an image in which some properties, such colour or brightness, differ from the areas surrounding those regions.

One of the first blob detectors was based on the Laplacian of Gaussian (LoG) [10, 11]. Mathematically, the LoG can be expressed as follows:

Given the already mentioned Gaussian function, the Gaussian scale-space representation $L(x,y;\sigma)$ of image $I(x,y)$ is the convolution between the image I and the Gaussian function

$$L(x,y;\sigma) = I(x,y) * G(x,y;\sigma)\ . \tag{2.5}$$

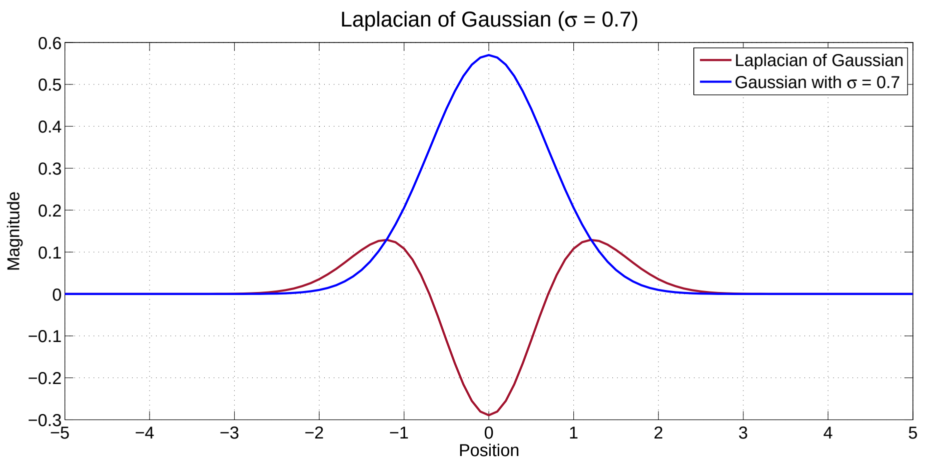


Figure 2.2: Laplacian of Gaussian (LoG) for $\sigma = 0.7$ - Operator in one dimension

Then, the Laplacian of Gaussian consists in applying the Laplacian operator,

$$\nabla^2 = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \ , \tag{2.6}$$

to the Gaussian scale-space representation $L(x, y; \sigma)$, i.e. $\nabla^2 L(x, y; \sigma)$. Equivalently, the LoG operator can be computed first,

$$\nabla^2 G(x, y; \sigma) = \frac{x^2 + y^2 - 2\sigma^2}{\pi\sigma^4} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \ , \tag{2.7}$$

and then this can be convolved with the image. The resulting element is called LoG scale-space representation. An example of a 1D LoG operator is shown in Figure 2.2.

In the literature [10, 11, 12, 13] the DoG is referred as an approximation of the LoG. Briefly explained, the Gaussian scale-space representation $L(x, y; \sigma)$ from (2.5) satisfies the diffusion equation $\partial^2 L = \frac{1}{2}\nabla^2 L$. Then, the Laplacian of the Gaussian operator $\nabla^2 L(x, y; \sigma)$ can also be computed as the limit case of the difference between two Gaussian smoothed images

$$\nabla^2 L(x, y; \sigma) \approx \frac{\sigma}{\Delta\sigma} \left(L(x, y; \sigma + \Delta\sigma) - L(x, y; \sigma - \Delta\sigma)\right) \ . \tag{2.8}$$

DoG is usually preferred over LoG due to its computational efficiency.

How is the DoG operator useful to find blobs? A blob-structure with certain size with respect to the standard deviation (σ) of the Gaussians will produce a strong response from the DoG operator. More precisely, for an image with black background and bright blobs, the DoG will produce a local minimum in the centre of the blob. For an image with bright background and dark blobs, the DoG will produce a local maximum.

The relation between the blob size and the standard deviation of both Gaussians might have several interpretations. The simplest one is to equal the blob radius to the point where the DoG operator crosses the x-axis, which mathematically is the point where the two Gaussians intersect. Then, when the convolution is being computed and the DoG operator is placed right over the blob, each pixel of the blob is multiplied by a negative value of the DoG, the closer pixels outside the blob are multiplied by a positive value of the DoG, and then all those values are summed up; thus, the resulting value will be locally minimum (or maximum) since the values for the convolution when the DoG is moved just one pixel away in any direction will be smaller (in absolute value) with respect to the previous one.

Even though this interpretation is correct, it might be not the best solution when the blob is not completely circular or when a cluster of blobs is present. In this case, the DoG operator would cover part of the blobs surrounding the blob to be detected, which would affect the output. Then, it is preferred that the whole DoG operator would cover only one blob. Kong *et al.* [11] suggest a different value

$$\sigma_{DoG} = (r-1)/3$$

for a blob with radius r. The x-axis cross-point is called here σ_{DoG}. This is motivated by the "3σ" rule that 99% of the energy of a Gaussian is concentrated within three standard deviations of its mean. Nonetheless, any DoG of a single scale will fail to detect blobs of different sizes.

To obtain the σ values, the Gaussian functions are equalled,

$$\frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left(-\frac{x^2}{2\sigma_1^2}\right) = \frac{1}{\sqrt{2\pi\sigma_2^2}}\exp\left(-\frac{x^2}{2\sigma_2^2}\right), \tag{2.9}$$

which makes

$$x = \sigma_{DoG} = \sqrt{ln\left(\frac{\sigma_1}{\sigma_2}\right)\left(\frac{2\sigma_1^2\sigma_2^2}{\sigma_1^2-\sigma_2^2}\right)}. \tag{2.10}$$

Thus, the approach is to fix σ_1 (the Gaussian with the smallest σ) and to calculate σ_2 based on the desired σ_{DoG}. This problem cannot be solved analytically, so a numerical method is necessary. This can be done in several ways. In this project, the Newton's method [14] was used.

The only pre-requisite is to establish a proper σ_1 and a good first approximation of σ_2. This should be: $\sigma_1 < \sigma_{DoG} < \sigma_2$. In this project, it was decided to fix $\sigma_1 = \sigma_{DoG} - \sigma_{DoG}/5$ and the initial approximation of $\sigma_2 = \sigma_{DoG} + \sigma_{DoG}/5$.

2.2 Watershed Segmentation

This segmentation method was first introduced by Lantuéjoul & Beucher in 1979 [15], although the mathematical definition was properly proposed in the beginning of the 90s by several scientists [16, 17, 18].

The general concept can be described as follows: A grey-scale image can be interpreted as a topographic relief if the magnitude of the image gradient in each pixel is

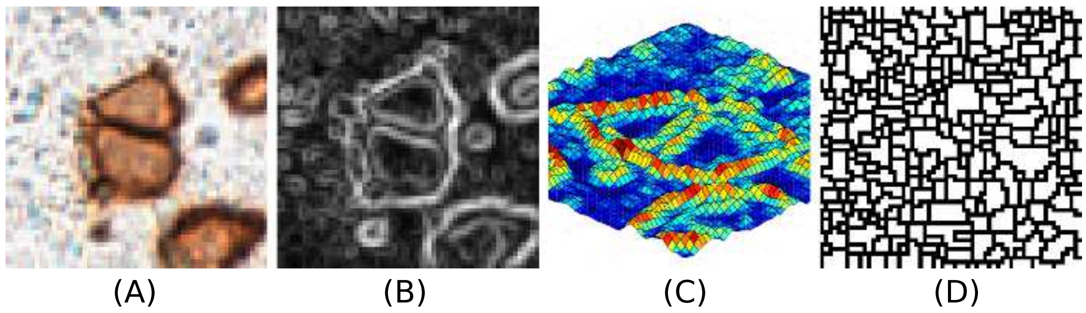


Figure 2.3: **(A):** A colour image. **(B):** The gradient magnitude of its equivalent grey image. **(C):** A representation of the gradient magnitude as a topographic relief (3D). **(D):** The resulting watersheds. Each white area is a catchment basin, and each black line is a watershed.

interpreted as its altitude in the relief (this is illustrated in Figure 2.3 - A, B & C). By the law of gravity, if a drop of water is imagined to fall on this topographic surface then it would follow the steepest path of descent until it reached a minimum point. Using the geographical terminology, the set of all possible paths that a drop of water could follow to reach a given minimum point defines a catch basin. Then, the watersheds of this surface correspond to the limits of the adjacent catch basins (Figure 2.3-D).

This first definition contains a major problem: in digital images it is possible that the direction of flow at a given pixel cannot be determined; for instance, a pixel surrounded by others with the same gradient magnitude. Thus, the method was conceptually modified in the 90s by doing a simple change: the topographic relief is imagined to be flooded by water coming out from various sources in the ground rather than from falling water. This version was called 'watershed by flooding'. In the last decade the improvements in this algorithm have come mainly from a time-complexity point of view [19].

Since usually the gradient image is quite irregular, with lots of minimum points, the algorithm tends to produce over-segmented solutions (Figure 2.3-D). Nonetheless, many pre- and post-processing techniques can be applied to reduce them: smooth or modify the gradient magnitude, join catchment basins based on statistical parameters, etc.

Other approaches use specific marker positions as sources of the flooding which have been either explicitly defined by the user or determined automatically with morphological operators or other ways. Newer techniques include stochastic processes to the basic method to determine the relevant watersheds [20], or the inclusion of learning-based approaches [21].

2.3 Morphological Operators

Morphological operations (ψ) process images based on shapes. They apply a *structuring element* ($\mathcal{B}$) to an image ($\mathcal{A}$) to generate another image. The basic operations are dilation and erosion.

The structuring element ($\mathcal{B}$) is usually a square or circle, and is expressed with respect to a local origin $\mathcal{O}$, called anchor point, which usually is the centre of the element.

To apply the morphological transformation $\psi(\mathcal{A})$ to the image $\mathcal{A}$ means that the structuring element $\mathcal{B}$ is moved systematically across the entire image. Assume that $\mathcal{B}$ is positioned at some point in the image; the pixel in the image corresponding to the anchor point $\mathcal{O}$ is called the *current* pixel. The result of the relation between the image $\mathcal{A}$ and the structuring element $\mathcal{B}$ in the current position is stored in the output image in the current image pixel position.

In this work, we will use binary structuring elements and binary images. Then, the basic morphological operations can be described as follows:

In *dilation*, the maximal pixel value overlapped by $\mathcal{B}$ will be stored in the output image in the current image pixel position. The resulting image is then a dilated version of the original one. Dilation is mathematically expressed with the symbol $\oplus$.

In *erosion*, the minimal pixel value overlapped by $\mathcal{B}$ will be stored in the output image in the current image pixel position. The resulting image is then an eroded version of the original one. Erosion is mathematically expressed with the symbol $\ominus$.

Both operations are illustrated in Figures 2.4 (dilation) and 2.5 (erosion), where the red blocks symbolize a binary pixel with value 1, the light brown pixels symbolize a binary pixel with value 0, and the dark red in the structure indicates the anchor point.

Finally, closing and opening are based on the previous operations:

Closing is defined as: $A \bullet B = (A \oplus B) \ominus B \quad \rightarrow$ Dilation followed by erosion.
Opening is defined as: $A \circ B = (A \ominus B) \oplus B \quad \rightarrow$ Erosion followed by dilation.

Closing is generally used to fill holes inside an object, whereas opening is used to remove small objects in an image.

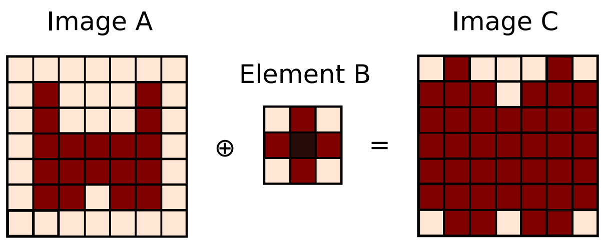


Figure 2.4: Graphic representation of the morphological operation *dilation*.

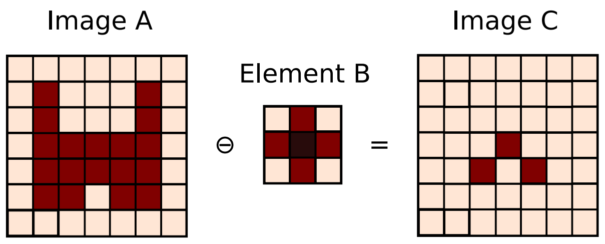


Figure 2.5: Graphic representation of the morphological operation *erosion*.

2.4 Support Vector Machines (SVM)

Support Vector Machines (SVM) is a supervised algorithm for classification and regression analysis, which uses learning models to analyse data and recognise patterns [22, 23, 24]. This means that this method has been given a set of labelled examples, each one belonging to one of two categories, and based on them SVM builds a model that can be used to classify new examples. SVM is a non-probabilistic classifier. However, some techniques have been developed to provide a ranking value (rank-SVM) or a probabilistic value (Platt's implementation [25]).

An SVM model is a representation of the examples as points in a space (called input or feature space), mapped in such way that is is possible to 'visualize' a gap between both categories. SVM builds a decision boundary, called hyperplane, in that gap. New examples are then mapped into that same space and predicted to belong to a category based on which side of the hyperplane they fall on. Ideally, that gap is clear and wide enough, which gives some certainty that new examples will be classified correctly. Usually, the categories are not completely separable, so SVM builds a hyperplane that minimizes the classification error of those training examples.

A simple example of how a hyperplane looks like is illustrated in Figure 2.6. In this case, two characteristics (features), colour and size, describe the elements, and the collection of known elements constitutes the training set. If those training examples are put in a space where one axis indicates the size and the other the colour, it is possible to draw multiple lines that can separate both classes (see Figure 2.6-left). However, selecting the best hyperplane is not a trivial task; in fact, it is the core of SVM.

Intuitively, it can be argued that if a line passes too close to the elements of one category with respect to the other, then it would be a noisy hyperplane because a new element from the first category could be classified easily as the second one. Thus, the best hyperplane would be the one that lies somewhere in the middle of the gap. To be specific, the goal is to find a hyperplane that gives the largest minimum distance to the training examples (γ). Twice that distance is called *margin* (2γ) (see Figure 2.6-right), and thus the optimal hyperplane maximizes the margin of the training data.

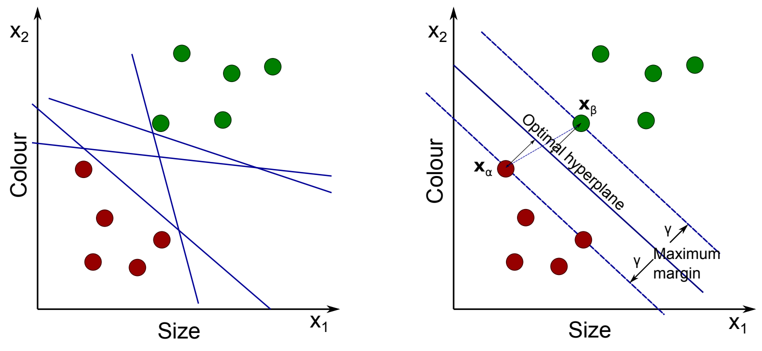


Figure 2.6: Left: Simple SVM space where multiple planes that can separate both categories have been drawn. **Right:** The optimal hyperplane

2.4.1 Computing the Hyperplane

Each element in the space is represented with its feature vector $\mathbf{x} = [x_1, x_2]^T$ and it belongs to one category $t = \pm 1$. A feature space can have infinite dimensions, but for explanation purposes only two dimensions will be used herein. Then, a straight line in a two dimensional space can be defined as

$$\mathbf{w}^T\mathbf{x} + b = 0 \; . \tag{2.11}$$

In order to solve the problem it is only necessary to obtain $\mathbf{w}$ and b. Then, the classification of new elements can be done by checking this:

$$\begin{aligned} &\text{if} \quad \mathbf{w}^T\mathbf{x}_{new} + b > 0 \quad \text{then} \quad t_{new} = +1 \\ &\text{if} \quad \mathbf{w}^T\mathbf{x}_{new} + b < 0 \quad \text{then} \quad t_{new} = -1 \end{aligned} \tag{2.12}$$

To obtain $\mathbf{w}$ and b, the margin (2γ) needs to be maximized, which is exactly the perpendicular distance from the decision boundary to the closest points on each side (see Figure 2.6-right). Knowing that $\mathbf{w}^T\mathbf{x}_\alpha + b = -1$ and $\mathbf{w}^T\mathbf{x}_\beta + b = +1$, the margin is

$$2\gamma = \frac{1}{||\mathbf{w}||}\mathbf{w}^T(\mathbf{x}_\beta - \mathbf{x}_\alpha) \quad \longrightarrow \quad \gamma = \frac{1}{||\mathbf{w}||} \; . \tag{2.13}$$

Thus, $\gamma = \frac{1}{||\mathbf{w}||}$ should be maximized. This is equivalent to minimizing $||\mathbf{w}||$, or also minimizing $\frac{1}{2}||\mathbf{w}||^2 = \frac{1}{2}\mathbf{w}^T\mathbf{w}$. There are some constraints, which are essentially that the points on one side and the other of the hyperplane should have the correct labels,

$$\begin{aligned} &\text{for} \quad \mathbf{x}_n \quad \text{with} \quad t_n = +1 \quad : \quad \mathbf{w}^T\mathbf{x}_n + b \geq +1 \\ &\text{for} \quad \mathbf{x}_n \quad \text{with} \quad t_n = -1 \quad : \quad \mathbf{w}^T\mathbf{x}_n + b \leq -1 \end{aligned} \tag{2.14}$$

Finally, all of this have been summarized to an optimization problem

$$\underset{w}{\text{argmin}} \, \frac{1}{2}\mathbf{w}^T\mathbf{w}, \qquad \text{subject to:} \quad t_n\left(\mathbf{w}^T\mathbf{x}_n + b\right) \geq 1 \; . \tag{2.15}$$

The constraints can be put inside the minimisation using Lagrange multipliers,

$$\underset{w}{\text{argmin}} \, \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{n=1}^{N} \alpha_n \left(t_n\left(\mathbf{w}^T\mathbf{x}_n + b\right) - 1\right), \qquad \text{subject to:} \quad \alpha_n \geq 0 \; . \tag{2.16}$$

Solving this equation (called primal optimization problem) is quite complex. Hopefully, it is possible to change it to a maximisation problem (called dual opt. problem),

$$\begin{aligned} &\underset{\alpha}{\text{argmax}} \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n.m=1}^{N} \alpha_n\alpha_m t_n t_m \mathbf{x}_n^T\mathbf{x}_m \; , \\ &\text{subject to:} \quad \sum_{n=1}^{N} \alpha_n t_n = 0, \quad \alpha_n \geq 0 \; . \end{aligned} \tag{2.17}$$

How the primal solution is changed to the dual solution and how this can be solved by quadratic programming require a long explanation, and this thesis does not have that objective.

In summary, in the new problem only the Lagrange multipliers are unknowns. Those multipliers, called α_n, indicate how much a training point affects the creation of the hyperplane. In fact, only the closest points will define the margin, and only those will have an α with a non-zero value. Those points are called *support vectors*, and give name to the algorithm. The classification of new elements can be expressed now as

$$t_{new} = sign\left(\sum_{n=1}^{N} \alpha_n t_n \mathbf{x}_n^T \mathbf{x}_{new} + b\right) \ . \tag{2.18}$$

Thus, it does not matter if the training data have thousands of points, that if only a few training points are support vectors, only their values are used. The computational time to classify new points is then improved extraordinarily.

2.4.2 Soft and Hard Margins

When there is an outlier in the training set, an SVM model will generate a hyperplane that does not separate appropriately both categories (see Figure 2.7-left). This is called *hard margin*. The solution involves relaxing the constraints. Then, the dual solution will now have this form

$$\begin{aligned} &\underset{\alpha}{\text{argmax}} \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n.m=1}^{N} \alpha_n \alpha_m t_n t_m \mathbf{x}_n^T \mathbf{x}_m \ , \\ &\text{subject to:} \quad \sum_{n=1}^{N} \alpha_n t_n = 0, \quad C \geq \alpha_n \geq 0 \ . \end{aligned} \tag{2.19}$$

which is exactly the same as (2.17) but with only an upper-bound in α_n. This is called *soft margin* (see Figure 2.7-right). C is one of the parameters that should be tuned every time

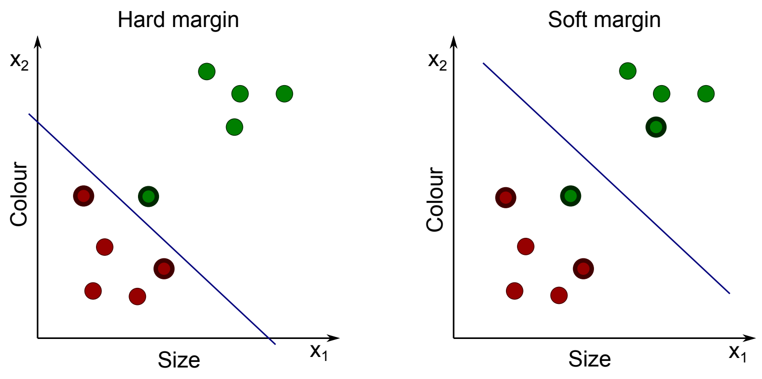


Figure 2.7: Left: SVM with hard margin. **Right:** SVM with soft margin - **Both**: Points with thicker borders are support vectors.

an SVM problem is analysed. In MATLAB it is called '*box-constraints*' since C keeps the allowable values of the Lagrange multipliers α_n in a "box", a bounded region. Increasing C means that the optimization attempts to make a stricter separation between classes (similar to *hard margin*), although it might over-fit the noise. Equivalently, reducing C towards zero makes misclassification less important, but it might under-fit the data.

In order to choose the best C, a cross-validation method might be used. In this project, a 10-fold cross-validation method was implemented. This entails that the training set is divided in 10 folds, and each time 9 folds are used for building the model and the remainder to validate the results. The final error rate is then the mean between all the error rates. Thus, the best C is the one that minimizes the error.

Besides that, an independent validation set was also built in this project to corroborate the results from the cross-validation method.

2.4.3 Kernels

The original definition of SVM dealt only with linear hyperplanes. However, a linear plane is not always capable of separating elements from two categories (see Figure 2.8-left). Luckily, a new method called "kernel trick" was introduced into SVM. There, the feature space is changed so that a linear hyperplane is suitable. For instance, this simple concept can be observed in Figure 2.8-right where the new space is defined as

$$\phi(\mathbf{x}_n) = x_{n1}^2 + x_{n2}^2 \; . \tag{2.20}$$

The new optimization problem is then

$$\begin{aligned} &\underset{\alpha}{\operatorname{argmax}} \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n.m=1}^{N} \alpha_n \alpha_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \; , \\ &\text{where:} \quad k(\mathbf{x}_n, \mathbf{x}_m) = \phi(\mathbf{x}_n^T)\phi(\mathbf{x}_m) \; . \end{aligned} \tag{2.21}$$

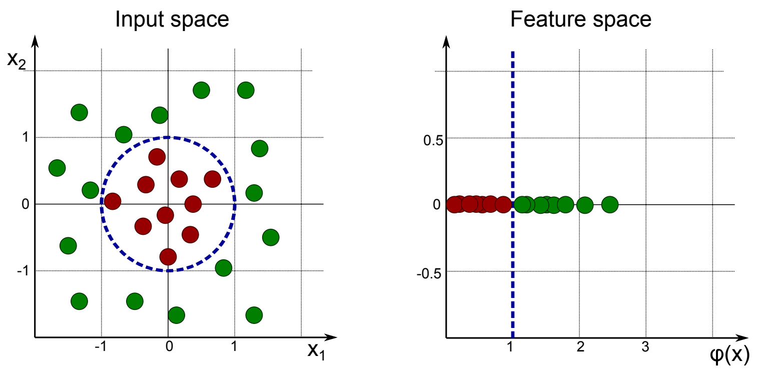


Figure 2.8: Left: (Input space) No linear plane can separate both classes - However, a circular one could. **Right:** (Feature space) Using the kernel trick it is possible to separate the classes with a line.

The interesting point here is that the algorithm is looking for linear boundaries but in some other space. This simple trick gave SVM a powerful tool to create extremely complex boundaries. Plenty of off-the-shelf kernels can be used. Besides the linear kernel, the most common ones are the Gaussian kernel

$$k(\mathbf{x}_n, \mathbf{x}_m) = exp\left(-\beta\left(\mathbf{x}_n - \mathbf{x}_m\right)^T\left(\mathbf{x}_n - \mathbf{x}_m\right)\right)\ , \tag{2.22}$$

and the polynomial kernel

$$k(\mathbf{x}_n, \mathbf{x}_m) = \left(1 + \mathbf{x}_n^T\mathbf{x}_m\right)^\beta\ . \tag{2.23}$$

In this project, besides the linear kernel, polynomial kernels of degree 2 (quadratic) & 3 (cubic) and a Gaussian kernel ($\beta = 1$) were tested.

2.4.4 Probabilistic Output from SVM

Platt proposed in 1999 [25] a method to transform the outputs of a classification model into a probability distribution over classes. In short, the method trains first an SVM model and then trains the parameters of an additional sigmoid function to map the SVM outputs into probabilities (the mathematical proof can be found in the original paper [25]). Platt's method has shown to be effective for SVM as well as for other types of classification models. This method will be used in the *flooding detector*.

2.5 Rank-SVM

Rank-SVM is a pairwise method based on SVM for designing ranking models. Even though Herbrich *et al.* started studying pairwise methods in 2000 [26], the algorithm was correctly proposed by Joachims in 2003 [27], and the original purpose was to improve the performance of an internet search engine. Chapelle and Keerthi [28] explain the concept from a more general perspective, and propose new methods to speed up the rank-SVM training.

Mathematically, this method creates a ranking model by minimising a regularized margin-based pairwise loss

$$\underset{\mathbf{w}}{\operatorname{argmin}}\ \frac{1}{2}||\mathbf{w}|| + \sum_{(i,j)\in P} l\left(\mathbf{w}^T\left(\mathbf{x}_i - \mathbf{x}_j\right)\right)\ , \tag{2.24}$$

where l is a suitable loss function such as $l(t) = max(0, 1 - t)$, called *hinge loss.*

The set of pairs P can be built by comparing hypotheses in such way that a pair $(i,j) \in P$ involves that hypothesis i is preferred over hypothesis j.

If this problem is further developed, it will be clear that it is equivalent to the classification SVM on pairwise difference $\mathbf{x}_i - \mathbf{x}_j$. This means that any SVM solver can be used to train a rank-SVM model, where $\mathbf{x}_i - \mathbf{x}_j$ is the positive class, and $\mathbf{x}_j - \mathbf{x}_i$ is the negative class. Furthermore, the projections of new elements onto $\mathbf{w}$ will provide their ranking order, which is equivalent to say that the ranking value of new elements will be the signed distance from them to the hyperplane.

It is important to highlight that *rank-SVM does not classify the elements*, it only ranks them, i.e. the side of the hyperplane is irrelevant here. Therefore, if a classification is also wanted, it is necessary to use a threshold obtained from a ranked training set to classify the new elements. This is the approach used in this project.

Why do we use rank-SVM? Since several hypotheses might lie inside the same cell, there is a need to choose the best one. Thus, it is necessary to rank or evaluate the hypotheses, not only to classify them.

We seek a method that takes into account not only the differences between good and bad hypotheses but also the differences among good hypotheses, knowing specifically when an element is better than other.

Platt's method just fits a logistic regression model using the classified elements of the training set, which means that the method looks in the feature space of the elements and fits them in a sigmoid based on their positions (similarities) relative to each other. This involves that if a large amount of positive elements are very close to each other in the feature space, the method considers them the 'best' elements. The question is: *Are they really the best ones?* That is the assumption that Platt's method makes.

On the other hand, rank-SVM model is built based on the explicit differences of pairs of hypotheses, and it will rank new elements based on those differences. In some way, the model has built a 'ranking feature space' where new elements are ranked based on their position. Therefore, this technique provides a more trustworthy method for solving our problem.

Nevertheless, for rank-SVM it is crucial to build a good training set where proper comparisons have been made: it should be clear that an element is better than another.

For this project a training set of 11 000 pairs was built. The objective was to create three different kind of pairs since the hypotheses could be classified into three categories: good, fair and bad. In this sense, two good-hypotheses or two bad-hypotheses were never compared between each other. The pairs could be good-fair, good-bad, or fair-bad (see Figure 2.9). Then, if one element was not clearly better than another, that pair was not included in the training set.

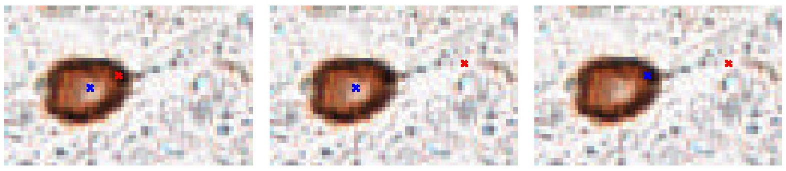


Figure 2.9: The three different kind of pairs that will be used for rank-SVM training. Those are: **Left:** Good-Fair. **Centre:** Good-Bad. **Right:** Fair-Bad.

2.6 Assessing a Classifier Performance

There are several ways to assess how good a classifier is. In this section only the methods used in this project will be explained.

The first and simplest one is called "**0/1 loss**", referred in this thesis as "mean error rate", without specifying the type of error. It is defined as the proportion of times the classifier is wrong. Mathematically it is defined as

$$\frac{1}{N}\sum_{n=1}^{N}\delta\left(t_n \neq t_n^*\right) \ , \tag{2.25}$$

where $t_1, ..., t_n$ are the set of predictions and $t_1^*, ..., t_n^*$ are the set of true labels ($\delta(a)$ is true if a is true and 0 otherwise).

A more complex method involves taking into account the class imbalance. Four quantities are necessary:

- **True positives** (TP): The no. of objects with $t_n^* = 1$ that are classified as $t_n = 1$.
- **True negatives** (TN): The no. of objects with $t_n^* = 0$ that are classified as $t_n = 0$.
- **False positives** (FP): The no. of objects with $t_n^* = 0$ that are classified as $t_n = 1$.
- **False negatives** (FN): The no. of objects with $t_n^* = 1$ that are classified as $t_n = 0$.

Based on those four quantities, several statistical measures can be computed. The most common are:

- **Sensitivity** (SE) or **True Positive Rate** (TPR): The ability to identify a condition correctly. Defined as:
$$SE = TP/P = TP/(TP + FN)$$
- **Specificity** (SP) or **True Negative Rate** (TNR): The ability to reject a condition correctly. Defined as:
$$SP = TN/N = TN/(FP + TN)$$
- **False Positive Rate** (FPR): Defined as:
$$FPR = 1 - SP = FP/(FP + TN)$$
- **False Negative Rate** (FNR): Defined as:
$$FNR = 1 - SE = FN/(TP + FN)$$
- **Precision** (PPV): Defined as:
$$PPV = TP/(TP + FP)$$

The *flooding detector* will use the sensitivity and specificity to evaluate the features. It is desirable that both have high values, although increasing one measure normally involves reducing the other.

However, the *rank-SVM detector* will use a different method since a threshold will be employed. For that purpose, the **Receiver Operating Characteristic** (**ROC**) curve will be used, which compares the sensitivity (True Positive Rate) with the False Positive Rate (FPR) [29]. For clarification purposes, two examples are depicted in Figure 2.10. The closer a curve is to the top-left corner, the better; in that point $SE = SP = 100\%$.

In order to compare classifiers it is preferred to reduce the ROC performance to a single scalar value. A common measure is to compute the *area under the ROC curve* (AUC). Since the AUC is a portion of the area of the unit square, its value can be between 0 and 1, where 1 represents the perfect classifier.

One of the properties of the AUC is that it is equivalent to the probability that the classifier will rank a randomly chosen positive element higher than a randomly chosen negative element.

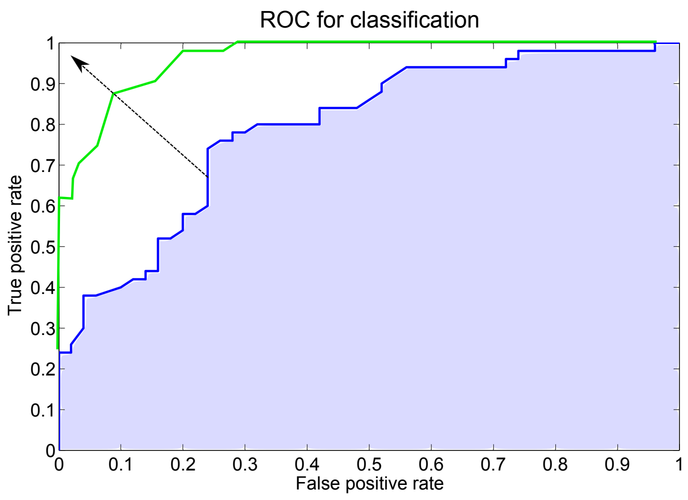


Figure 2.10: Example of two ROC curves (blue and green). A better classifier will have a curve which will tend towards the top-left corner (just as the arrow indicates). Thus, the green classifier is better than the blue. The area under the curve (AUC) for the blue classifier (the bluish area) is 0.7918.

2.7 Maximum Weighted Independent Set Problem

An Independent Set (IS) problem can be defined as follows: Given a graph $G = (V, E)$, we say a set of nodes $S \subseteq V$ is independent if no two nodes in S are joined by an edge [32]. It is simple to find small independent sets in a graph; for example, a single node forms an independent set. However, finding the largest independent set is a hard task. This problem belongs to the NP-hard problems (Non-deterministic Polynomial-time hard), which means that there are not known algorithms that can solve it in polynomial time. The IS problem is illustrated in Figure 2.11-left, where the maximum independent set contains the nodes {1, 4, 5, 6}.

The Weighted Independent Set (WIS) problem is a version of the IS problem where the nodes possess a value or weight. Thus, the maximum WIS solution is an independent set with maximum total weight. This is illustrated in Figure 2.11-right, where the maximum weighted independent set contains the nodes {2, 3, 7} with a total weight of 6. Note that the IS problem is just a WIS problem where all the weights are one.

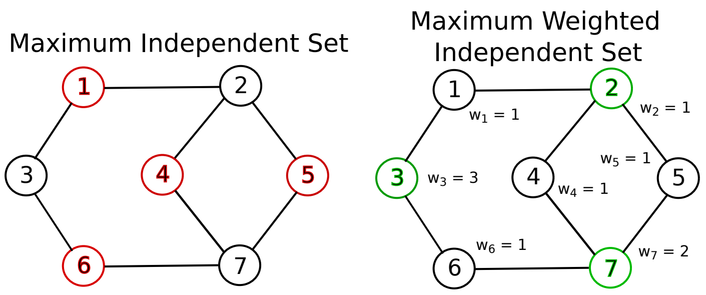


Figure 2.11: Left: A graph with 7 nodes and 8 edges whose largest Independent Set has a size of 4 (nodes in red). **Right:** A graph with 7 nodes with weights and 8 edges whose largest Weighted Independent Set has a size of 3 (nodes in green).

3

Algorithm Outline

Two different detectors will be developed, and each one approaches the problem in a different fashion. The first one, called '*rank-SVM detector*', only provides detection, whereas the second one, called '*flooding detector*', also provides segmentation.

The first approach was the original design, and the second one was born after a brainstorming process to attempt to overpass the limitations that the first approach faced.

Both algorithms have a similar structure. This structure consists of a first detector, a feature extractor, a support vector machine classifier, and a final selector of hypotheses. Apart from that, the flooding algorithm includes a block for segmentation. The sketches for both designs are shown in Figure 3.1 and Figure 3.2.

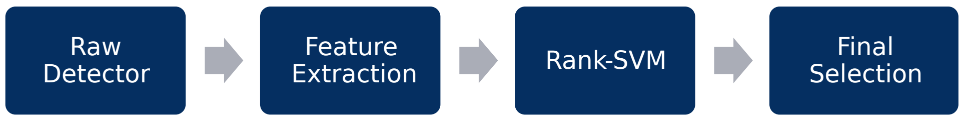


Figure 3.1: Algorithm sketch 01 - A Rank-SVM algorithm for cell detection

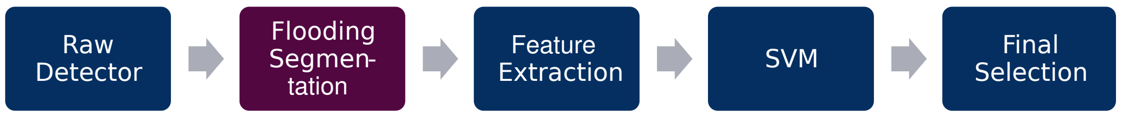


Figure 3.2: Algorithm sketch 02 - A flooding algorithm for cell detection

The objectives of these blocks are:

- **Raw detector**: It provides the position of hypothetical cell centres. These positions are called hypotheses. Each hypothesis can be true or false, and the rest of the algorithm will attempt to classify them correctly. The detector is based on the difference-of-Gaussians algorithm. This block is common for both approaches.

- **Flooding segmentation**: It provides one or more segmentations for each hypothesis. Then, the hypotheses would be redefined as a point in the image (from the detector) plus a segmentation. A novel method based on the watershed algorithm is used.

- **Feature Extraction**: It obtains a set of feature descriptors from each hypothesis. Those features are different depending on the approach. For the rank-SVM algorithm, they are:

  - *Gradient feature*: Histogram of Gradients (HoG).
  - *Colour features*: Colour histogram, mean colour, and standard deviation.

  For the flooding algorithm, the features are:

  - *Colour features*: Mean colour & standard deviation, and mean colour & standard deviation after certain erosion of the segmentation.
  - *Size-dependent shape features*: Area, perimeter, thickness, convex area, and convex perimeter.
  - *Size-independent shape features*: Formfactor, elongatedness, convexity, and solidity.

- **Support Vector Machine (SVM) and rank Support Vector Machine (rank-SVM)**: It classifies the hypotheses as true or false cells based on the features extracted. Both methods are intrinsically related, being the rank-SVM a posterior variation of SVM which can rank the hypotheses, providing a score for each hypothesis.

- **Final selection**: It obtains the combination of true hypotheses that fulfil a required rule. In this case, the rule is to maximize a total score. It solves the problem by using a maximum weighted independent set approach.

The core of the algorithm lies in the SVM classifier and the feature extraction, and the rest of the blocks just aim to simplify or refine the method.

The *raw detector* aims to speed up the process by just selecting a reduced number of pixels of the image as potential cell centres. All pixels of the image could have been considered hypotheses, but this would have increased the time to solve the problem unnecessarily.

The *final selector* aims to refine the solution by just selecting the most appropriate positive-hypotheses that are compatible between each other. This is not a trivial task since we do not want two cell detections within one cell.

Regarding the segmentation block, it also helps to obtain a better detection. First, it gives the classifier a new set of features to work with. Second, it simplifies the problem in the *final selector* even more, using the segmentation to avoid incompatibilities between cell detections.

4

Common: Raw Detector

This chapter describes the first block in the algorithm structure: the *raw detector*. This block is common for both the *rank-SVM detector* and the *flooding detector*.

The *raw detector*, which is based on the difference of Gaussians (Chapter 2), aims to obtain a balance between these two objectives:

1. Each cell in the image contains at least one hypothesis.

2. The number of hypotheses in the whole image should be as small as possible.

Here a hypothesis refers to a position in the image that is potentially a cell centre. The best hypothesis would be the one that is the closest to the real centre of a cell. Nonetheless, a hypothesis near the inner borders of a cell could also be considered true if no better hypothesis can be generated. Figure 4.1 shows some true and false hypotheses.

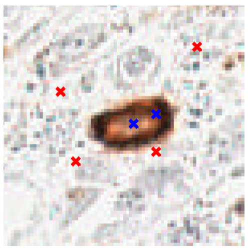


Figure 4.1: Visual definition of true hypotheses (blue) and false hypotheses (red)

4.1 Methodology

The images of this project contain slightly irregular cells in shape and colour. Even though they tend to be circular, elongated shapes are also common, and their colour intensity is not uniform, with borders usually darker than the centres. The average cell has a radius of 10 pixels, although cells with larger (up to 15 pixels) and smaller radius (down to 6-7 pixels) are also present.

So-called "weak cells" are also visible, which look like cells that were not properly stained in the immunohistochemical process, appearing partially transparent. Those cells were partially cut when the tissue section was obtained, and then less surface was available to react to the staining. It was not clear whether those cells should be considered 'positives'. Eventually, as the objective was defined, it was decided to detect every cell-like structure.

Therefore, due to the presence of cells of different sizes and shapes, each one will yield a local maximum in its centre for a DoG kernel with the right standard deviation (see Figure 4.2). Our experiments showed that $\sigma_{DoG} = 3 = (r_{average} - 1)/3$ is a good choice for the majority of cells (see Figure 4.2 top-right). This agrees with Kong suggestion [11]. Not only this produces hypotheses at almost every cell but they are also well centred. $\sigma_{DoG} = 2$ is also a good choice that covers small cells not detected in the previous case (see Figure 4.2 top-left). Thus, our final solution was to use both standard deviations.

Another solution was considered to detect small cells: to use also the local minimum. Just as the local maxima detect blobs in these images, the local minima appear in spots with a darker colour than the surroundings. This is specially relevant in the edges of the cells (see Figure 4.2 bottom-right). However, while this solution improved the detection of small separated cells, it did not improve the overall results as it increased the risk of getting multiple detections within the same cell.

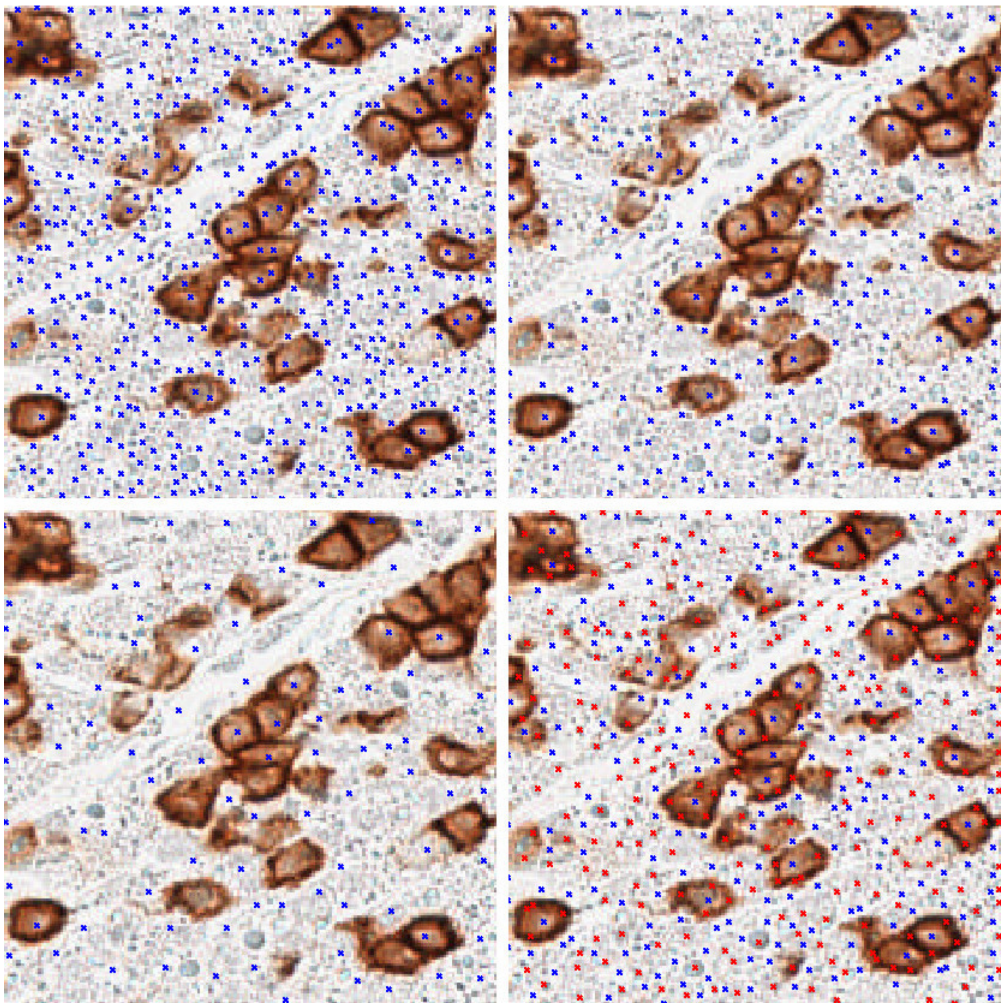


Figure 4.2: Local maxima points for a DoG detector with σ_{DoG} of: **Top-Left:** 2 pixels. **Top-Right:** 3 pixels. **Bottom-Left:** 5 pixels. **Bottom-Right:** Local maxima (blue) & minima (red) for $\sigma_{DoG} = 3$ pixels

5

Rank-SVM Detector: Feature Extraction & Evaluation

In the *rank-SVM detector*, hypothetical cell centres are generated using a difference-of-Gaussians detector (Chapter 4) and fed into a rank-SVM classifier (Chapter 2). This chapter will describe and evaluate the different features tested for the classifier.

In Section 5.1 the features will be described and the method to extract them will be explained. The evaluation of the independent features will be addressed in Section 5.2. Finally, the evaluation of the selected features as a single feature vector will be discussed in Section 5.3.

5.1 Feature Extraction

Two kinds of features were considered: **colour** and **gradient**. A mask surrounding the hypotheses will be used to extract them (Figure 5.2-left). Several masks were implemented. They are depicted in Figure 5.1. For each region of the mask a feature descriptor is computed, and the final feature vector is the concatenation of those descriptors.

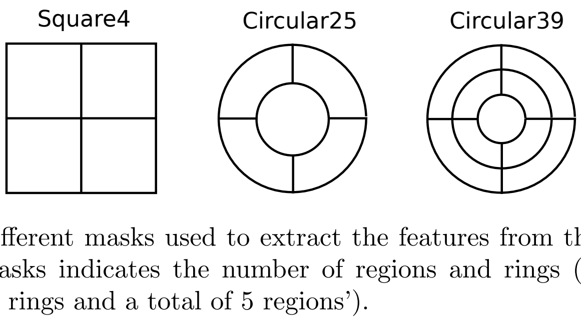


Figure 5.1: The different masks used to extract the features from the hypothesis-points. The name of the masks indicates the number of regions and rings (*circular25* means 'a circular mask with 2 rings and a total of 5 regions').

5.1.1 Colour Features

The colour representations are:

- *Colour histogram* (CH).
- *Mean colour* (MC).
- *Standard deviation of the colour* (STD).

Since our images are in colour, a feature value for each colour layer is computed, and the resulting feature vector is the concatenation of them. Two colour models were tested: HSV and RGB. This comes from a suggestion made by Angulo et al [20].

5.1.2 Gradient Feature: Histogram of Gradients (HoG)

This descriptor is based in the Histogram of Oriented Gradients, suggested by Dalal and Triggs [30], which describes the appearance and shape of an object in an image by using the distribution of intensity gradients.

The histogram of gradients can be obtained as follows:

- For the whole image, the gradient magnitude and orientation is computed by using $[-1, 0, 1]$ for horizontal and $[-1, 0, 1]'$ for vertical,
- In case of colour images, for each pixel the gradient with the largest magnitude among the three colours (RGB) is taken.
- The gradient orientation will be quantized to a certain number of bins, normally 8 or 9, which can be unsigned ($0° - 180°$) or signed ($0° - 360°$) (see Figure 5.2-centre).
- The histogram of gradients is calculated for each region of the mask, where the vote, which is the gradient magnitude, is interpolated between neighbouring bin centres (see Figure 5.2-right).
- The descriptor is finally normalized, which is based on the Euclidean distance.

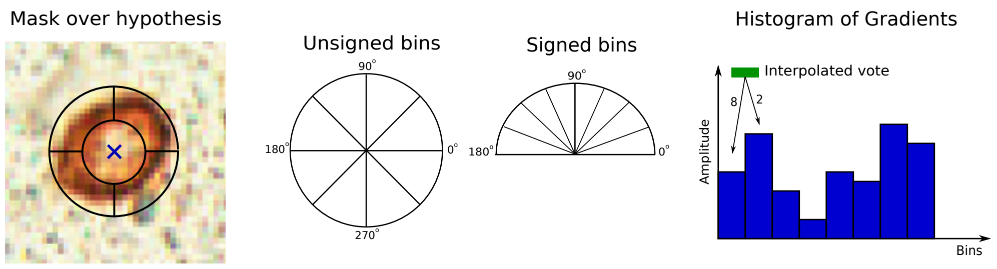


Figure 5.2: HoG descriptor. **Left:** How a mask is applied in a hypothesis. **Centre:** The two kind of gradient orientation in HoG. **Right:** How the gradient magnitude is interpolated in the bin distribution.

5.2 Feature Evaluation

What parameters should be tuned for each feature?

- Type of mask.
- Size (radius) of the mask.
- The SVM kernel and the box-constraint (C).
- Colour-model of the image: RGB or HSV.
- For histogram features (CH and HoG), the number of bins.
- For HoG, also the orientation.

Each feature is tuned independently. This means that they do not have to share the same parameter values. On the other hand, since the colour feature has three feature representations, a thorough test will be performed in order to discover if those features are redundant, i.e. if using the three of them is beneficial or only one is enough.

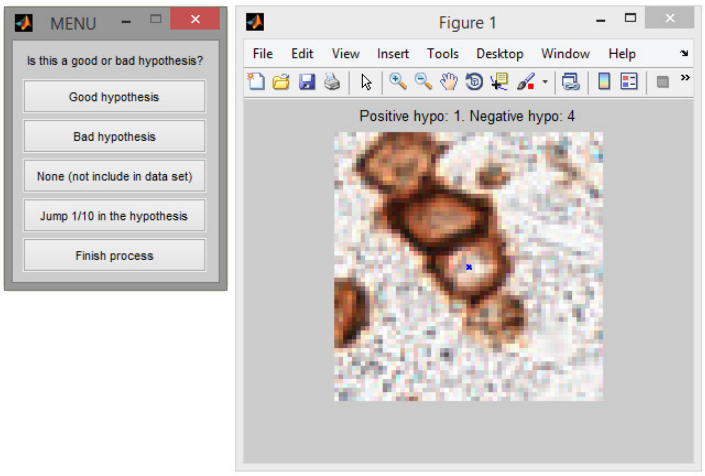
Four types of rank-SVM kernels were tested (linear, quadratic, cubic, and Gaussian $\beta = 1$). All of them provided practically identical results (each one tuned in its best C), being the linear kernel the one that performed slightly better. To simplify the chapter, only the results for the linear kernel will be displayed.

How are the parameters selected? As was explained in the theory chapter, rank-SVM does not classify elements but ranks them. Nonetheless, it is possible to provide classification if a threshold is employed on the ranking values. For that purpose, we require the following data sets:

- *Training set of pairs of hypotheses.* In this project, 11 000 pairs were computed.
- *Training set of single hypotheses.* In this project, 1000 true hypotheses and 7000 false hypotheses were used.
- *Validation set of single hypotheses.* Same as before.

The reason for the imbalance in the training and validation set of single hypotheses is due to the images used, where the background covered more area than the cells. To be specific, both sets were obtained by using the *raw detector* on two different images and then classifying manually the hypotheses. For that purpose, a simple script was written. This is depicted in Figure 5.3.

The process by which the best parameters were assessed is: For each feature with a certain combination of parameters (type of mask, size, etc.), the rank-SVM model is built using the training set of pairs of hypotheses. This provides the hyperplane for ranking. The training set of single hypotheses uses that hyperplane to obtain their rank value. Now we have a collection of known true and false hypotheses with their


MENU
Is this a good or bad hypothesis?
Good hypothesis
Bad hypothesis
None (not include in data set)
Jump 1/10 in the hypothesis
Finish process
Figure 1
File Edit View Insert Tools Desktop Window Help

Positive hypo: 1. Negative hypo: 4

Figure 5.3: Simple script was written to create the training set and validation set of single hypotheses. The hypotheses to be included in those sets should be clearly true or false, so an option to not include rare hypotheses was implemented.

scores. Thus, it is easy to use a threshold on those scores that separate true and false hypotheses. Instead of looking for a single threshold and its misclassification error rate, the area under the curve (AUC) from the ROC curve is employed (see Section 2.6). This provides a better understanding of how well the features perform. Then, if this is repeated for each combination of parameters, the best will be the one that maximizes the AUC. The validation set is just used to confirm the results.

Histogram of Gradients (HoG). This feature is the one containing more parameters to tune. A first evaluation of the ranking hyperplane was performed by employing a 10-fold cross-validation method on the *training set of pairs*. The results, depicted in Figure 5.4, show how the error rate depends on the mask and its orientation.

Note that this graph does not indicate the error rate of the classifier but how good the ranking is, which as a result will be related with the former one (this was later checked favourably). The use in that graph of signed HoG (360°) with 8 bins and unsigned HoG (180°) with 9 bins was a suggestion from [30]. Several conclusions can be made based on that graph:

- Signed HoG (360°) performs much better than unsigned HoG (180°). Thus, the unsigned version was tossed aside from the final design.
- The masks *circular25* and *circular39* originates an almost zero error rate, whereas the square one does not perform so well. In fact, for MC and CH features, the square mask behaves even worse. Thus, the square mask was discarded and no data for that mask will be displayed in the rest of the chapter.
- *Circular25* and *circular39* perform similarly around $radius = 12 - 14$. Thus, more tests need to be carried out to choose the most appropriate mask.

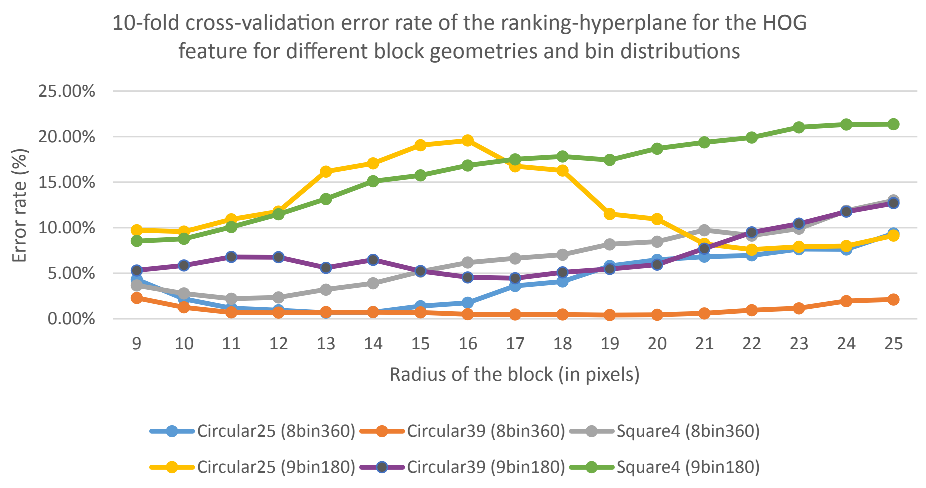


Figure 5.4: 10-fold cross-validation error rate performed in the ranking hyperplane (HSV).

This test was performed by using the HSV colour-map, even though Dalal and Triggs [30] suggest to use RGB. The reason behind this is that HSV provides, for identical parameter values, better results (see Table 5.1).

After studying the AUC for all the cases and combinations using the *validation set*, the following conclusions can be made:

- RGB colour model requires a very small C in the kernel. This usually means that both classes are quite mixed together in the feature space, which is unsatisfactory, so a less *hard* hyperplane performs better.
- The number of bins is not important as long as it is high enough. Ten bins was the final selection. Including more than that does not improve the results while it creates larger feature vectors (computationally more expensive).

The final results are shown in Table 5.1.

HoG Feature (signed orientation) - Best combination of parameters					
Colour-map	Mask	Radius	Bins	Kernel	AUC
RGB	*circular25*	12	10	Linear, $C = 5 * 10^{-4}$	0.9389
	circular39	16	10	Linear, $C = 5 * 10^{-3}$	0.9589
HSV	*circular25*	12	10	Linear, $C = 1$	0.9725
	circular39	16	10	Linear, $C = 0.1$	0.9771

Table 5.1: HoG feature: Best parameters for four different configurations and their AUC value for the validation set.

Mean Colour (MC). The AUC (see Table 5.2) shows that:

- All four variants yield a really good classification rate.

- For HSV, the best choice of radius for *circular39* is proportional to the best radius for *circular25*. This might indicate that the two inner rings in *circular39*, which have the same size as *circular25*, might be the ones doing the classification.

Mean Colour Feature - Best combination of parameters				
Colour-map	Mask	Radius	Kernel	AUC
RGB	*circular25*	15	Linear, $C = 1$	0.9973
	circular39	14	Linear, $C = 1$	0.9971
HSV	*circular25*	16	Linear, $C = 1$	0.9973
	circular39	23	Linear, $C = 0.05$	0.9969

Table 5.2: Mean Colour feature: Best parameters for four different configurations and their AUC value for the validation set.

Standard deviation (STD). The AUC (see Table 5.3) shows that this feature does a much better performance in HSV than in RGB.

Colour Standard Deviation Feature - Best combination of parameters				
Colour-map	Mask	Radius	Kernel	AUC
RGB	*circular25*	14	Linear, $C = 1$	0.9548
	circular39	22	Linear, $C = 1$	0.9597
HSV	*circular25*	9	Linear, $C = 1$	0.9892
	circular39	16	Linear, $C = 0.05$	0.9915

Table 5.3: Colour Standard Deviation feature: Best parameters for four different configurations and their AUC value for the validation set.

Colour Histogram (CH). The AUC (see Table 5.4) shows almost identical results. The pattern in the sizes and C might corroborate that the information is equally extracted using one colour model or the other.

Colour Histogram Feature - Best combination of parameters					
Colour-map	Mask	Radius	Bins	Kernel	AUC
RGB	*circular25*	16	10	Linear, $C = 5$	0.9968
	circular39	13	10	Linear, $C = 5$	0.9973
HSV	*circular25*	16	10	Linear, $C = 5$	0.9969
	circular39	13	10	Linear, $C = 1$	0.9970

Table 5.4: Colour Histogram feature: Best parameters for four different configurations and their AUC value for the validation set.

5.3 Final Selection of Features

Once each individual feature has been tuned to its best performance, the next step of the evaluation should be carried out: to find the best feature vector.

Several combinations were tested, and the results are given in Table 5.5 for the mask *circular25*. *Circular39* provided equal results. *Circular25* was preferred for our final design since the feature vector is significantly smaller, which would entail a better computational complexity.

Combinations of features		
Combination	Kernel	AUC
No 01 - circular25: HoG HSV - MC RGB - STD HSV - CH RGB	Linear, $C = 5$	0.997712
No 02 - circular25: All RGB	Linear, $C = 1$	0.996964
No 03 - circular25: All HSV	Linear, $C = 5$	0.997353
No 04 - circular25: Like No 1 with no CH	Linear, $C = 1$	0.996792
No 05 - circular25: Like No 1 with no MC	Linear, $C = 1$	0.997693
No 06 - circular25: Like No 1 with no HoG	Linear, $C = 1$	0.997334
No 07 - circular25: Like No 1 with no STD	Linear, $C = 5$	0.998069
No 08 - circular25: Only HoG HSV + CH RGB	Linear, $C = 0.5$	0.998013
No 09 - circular25: Only HoG HSV + CH HSV	Linear, $C = 5$	0.996973

Table 5.5: Combinations of features.

The conclusions based on these tests are:

- Standard deviation is a counter-productive feature, i.e. it is more beneficial not to include it (cases No. 1 vs No. 7). This is the only feature that generates such output.
- Using both colour features, MC and CH, seems unnecessary due to the small improvement (cases No. 7 vs No. 8). In this sense, CH performs better than MC, so this should be chosen.
- CH performs much better in RGB than in HSV when it is used next to HoG (cases No. 8 vs No. 9).

In summary, the final feature vector has the following form:

Histogram of Gradients (circular25 HSV) + Colour Histogram (circular25 RGB)

In Figure 5.5 the sensitivity and specificity of this design are depicted based on the *validation set*. It should be highlighted that the validation set had 7 times more false hypotheses than true hypotheses. This means that these values are just illustrative, i.e.

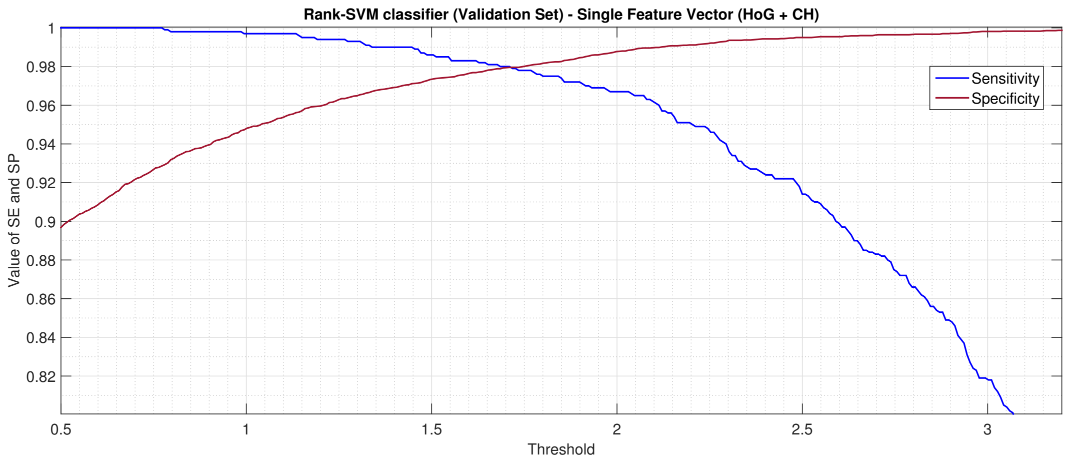


Figure 5.5: Single Feature Vector (HoG + CH): Sensitivity and Specificity based on the threshold for the validation set.

an image with a different ratio between true/false hypotheses would produce a slightly different graph.

In conclusion, both the sensitivity and the specificity are quite high. For our final design we believe it is advisable to set several thresholds so the possible user could make his own selection.

6

Flooding Detector: Segmentation

In the *flooding detector*, hypothetical cell centres are generated using a difference-of-Gaussians detector (Chapter 4). This chapter describes the segmentation method that is employed in each of those hypotheses. This method is based on the watershed algorithm, which was introduced in Chapter 2.

The motivation behind this approach is to provide an alternative to the lack of segmentation from the *rank-SVM detector*. This segmentation will turn to be helpful for the final selection of hypotheses. We will provide such a segmentation while keeping the original structure: the DoG raw detector, the SVM classifier, and the final selector.

The segmentation algorithm will be described in Section 6.1, followed by a brief discussion of the method in Section 6.2, and the illustration of the algorithm performance for two distinctive cases in Section 6.3.

6.1 Segmentation Algorithm

Based on the interpretation of the magnitude of the image gradient as a topographic relief and a set of sources (the point-hypotheses from the *raw detector*), each source will generate a flow of water (flood) independently, and once a source has produced its flow it will be removed. Therefore, no interactions between different sources will occur. The 'size' of that flood will be defined by the area to cover, which should be slightly higher than the area of the largest cell. In this case, a size of 300 pixels was used. At each iteration of the flooding process, the neighbouring pixel to the flood with the smallest altitude is included in the flood. If more than one pixel have the same minimum altitude, all of them are included at the same time. Every time a pixel is included in the flood, the ratio between the volume and the area occupied by the flood will be computed. This ratio will be called 'relative volume' from here onwards. The area is just defined as the number

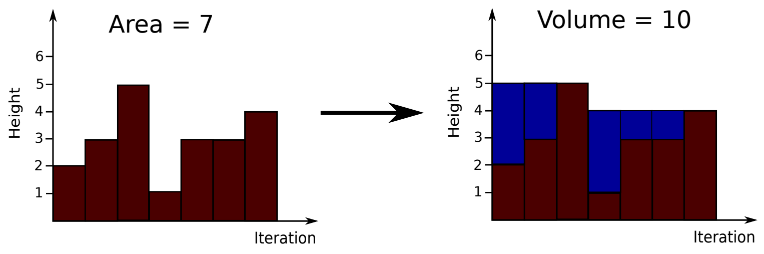


Figure 6.1: How the volume of water is computed based on the altitude graph. The area is just the number of pixels (iterations) involved. The water (blue blocks) in each pixel fill all the space until it reaches the highest altitude to its right side.

of pixels in the flood. The volume is computed based on the topographic relief; this can be obtained by tracking the altitudes in each iteration of the flood (Figure 6.1). Based on that, some stable points will be obtained, defined as local maxima of the relative volume in a range of 25 points. The first 30 iterations of the flooding process are not considered for that purpose. Every time a stable point occurs, the region covered by the flood will define a 'raw segmentation'. Thus, each source will produce as many segmentations as number of stable points. Finally, each raw segmentation will be processed with some morphological operations: First a closing operation with a circular structuring element (size 4 pixels) will be used to fill holes; later the *drop of water principle* will be applied to the closest pixels (so the pixels that are leaned towards the segmentation will be included in it); and finally an opening operation with a small circular structuring element (size 1 pixel) will be employed to erase possible protuberances in the edges.

If no stable segmentations can be generated from a hypothesis, it will be discarded. However, this part of the detector was not designed to reject hypotheses but to expand the information of them with the segmentations.

In order to produce better results, the original image is smoothed with a Gaussian filter of $\sigma = 1$ before computing the gradient magnitude. This value was chosen after several tests, which showed that any Gaussian filter with $0.5 \leq \sigma \leq 2$ is a good choice.

All the parameters in the algorithm can be expressed as a function of the size of the average radius. Thus, this is the only input value.

6.2 Discussion

The algorithm was conceived when we became aware that the flood generated by a source inside a cell shows a distinctive pattern: it starts filling the cell until it reaches certain point, called *spillage point*, from where the flood will spread outside the cell. This can be observed in Figure 6.4-top. Moreover, it was also noticed that the spillage point could easily provide a rather good segmentation of the cell if some morphological operations were applied. In a few cases the generated segmentation was far from the expected, but it was observed that imperfect segmentations could still be useful for our final goal, the detection.

An important issue to solve was to develop a robust method to find the spillage point for all cases, and this turned to be not a trivial task. Our tests showed that, for the majority of cells, the highest peak in the *relative volume* graph was the spillage point. For the other cases, that point was shadowed by a highest peak, which usually was the spillage point of a secondary cell. Thus, in order to solve the problem, it was decided to consider all the peaks in the relative volume graph that were locally maxima in a range of 25 points, which were called *stable points*. Doing so, the desired spillage point would practically always be included as one of the stable points. Then, we could make use of the last block, the *final selector*, to select the most adequate segmentation.

All these facts were tested extensively. To be precise, the 1000 positive points of the *training set of single hypotheses* from the *rank-SVM detector* were analysed, and only 40 cases could not generate any stable point. The remainder 960 cases generated their spillage point as one of the stable points. A 4% error was considered to be acceptable.

These are the benefits of this method:

- It is simple to implement.
- It produces a good segmentation for isolated cells.
- The *spillage point* can be found easily.
- If a cell contains more than one source, most probably all of them would converge to the same flooding pattern, providing the same *spillage point*, and so the same segmentation.

6.3 Illustrative Performance of the Algorithm for Two Different Scenarios

For illustrative purposes, two distinctive cases are depicted herein, showing two different patterns. The first one, called an '*isolated cell*', is defined as a cell completely surrounded by background. The second one, called '*a cell in a cluster*', is defined as a cell completely surrounded by other cells. All the steps, from the creation of the stable points until the application of the morphological operations, are depicted in Figure 6.4 for an isolated cell and in Figure 6.5 for a cell in a cluster. Their stable points are shown in Figure 6.2 and Figure 6.3, respectively.

Looking at Figure 6.4 it is clear that the algorithm creates a good segmentation for isolated cells. In this case, the best segmentation is obtained from the third stable point, which correspond to the spillage point. Nonetheless, any of the other two segmentations would be acceptable.

For cell clusters the problem is more complex. Looking at Figure 6.5 it is patent that the best (but not perfect) segmentation is the first one, and the others should be rejected since they affect other cells. A thorough discussion about how this affects the final solution will be given in Chapter 9.

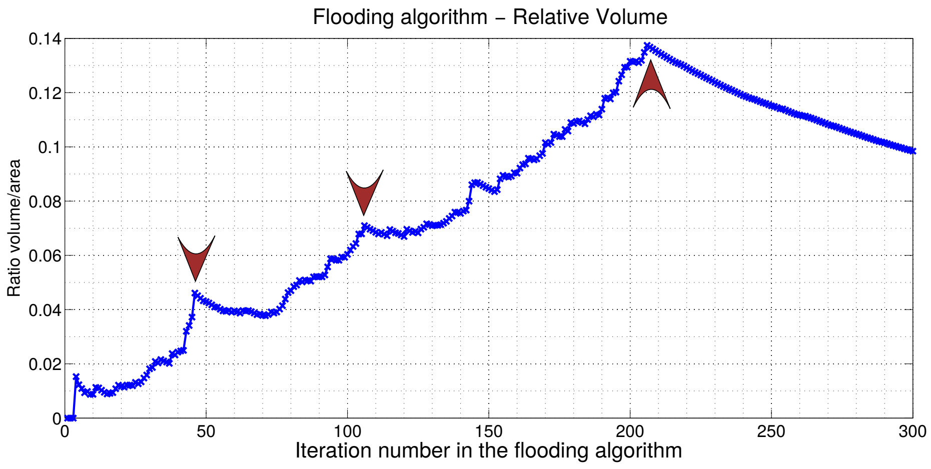


Figure 6.2: Relative volume graph for the isolated cell from Figure 6.4. The red arrowheads indicate the detected stable points.

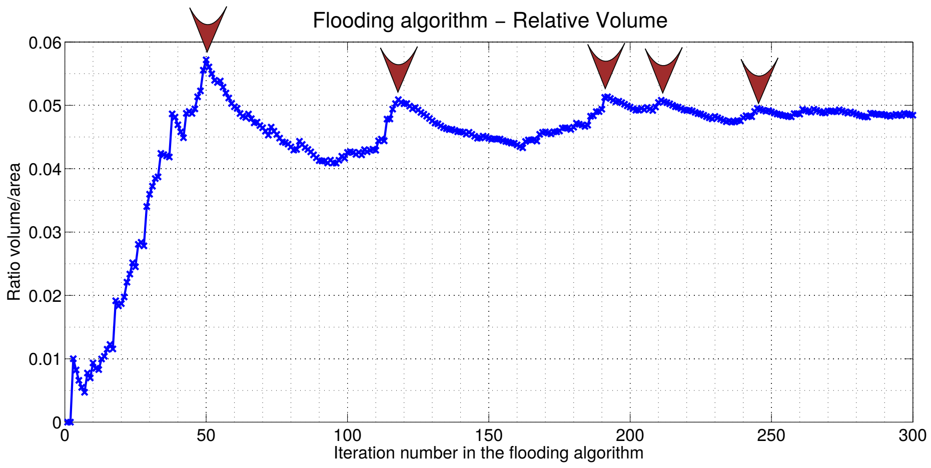


Figure 6.3: Relative volume graph for the cluster of cells from Figure 6.5. The red arrowheads indicate the detected stable points.

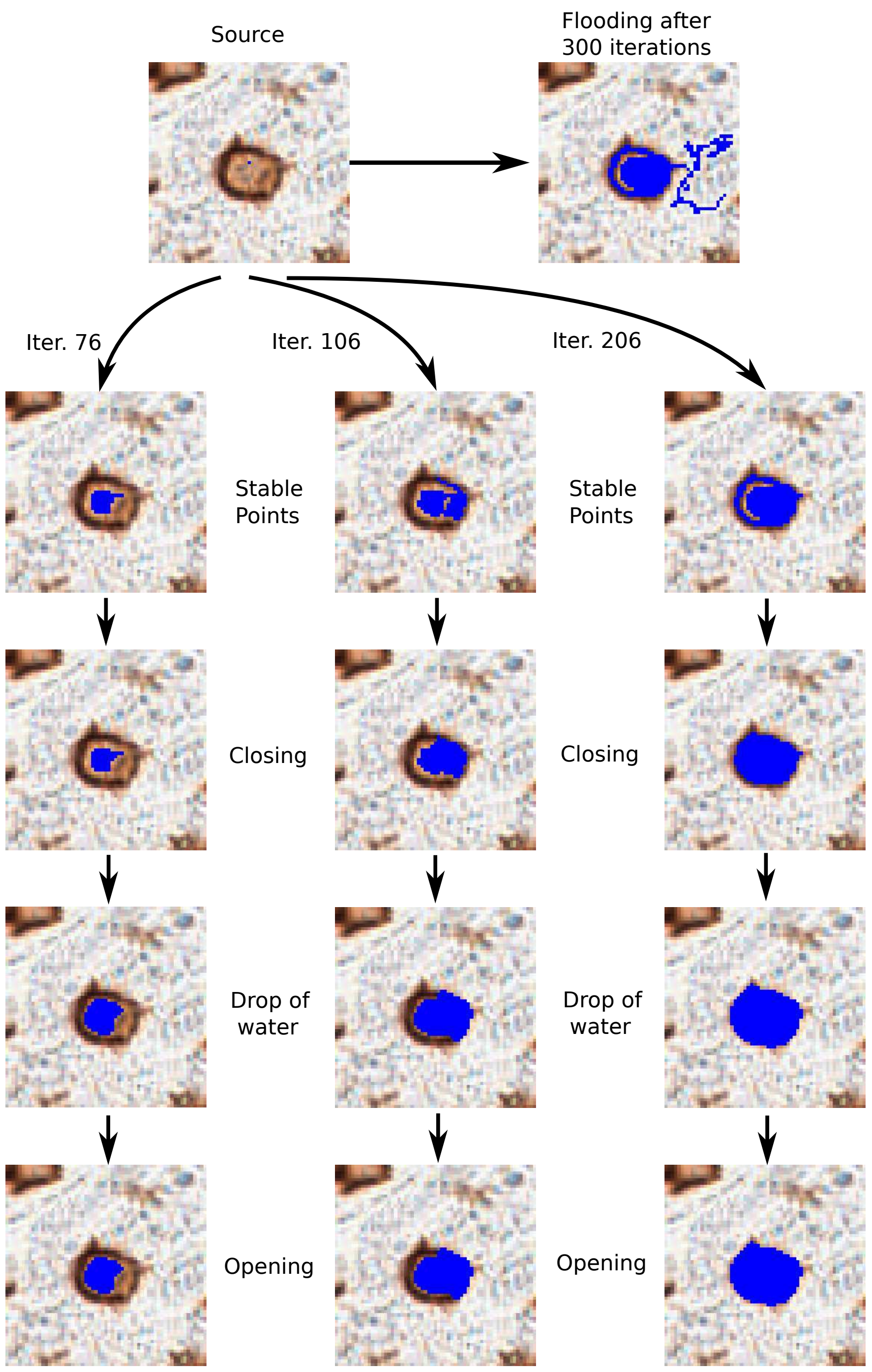


Figure 6.4: Whole process of the flooding segmentation for an isolated cell. The initial source in the cell originates a flooding pattern (top right) of 300 pixels long (called iterations), but only 3 stable points occur at iteration 76, 106 & 206. The raw segmentation (second row) at those stable points are processed by applying closing (third row), drop of water principle (fourth row) and opening (fifth row).

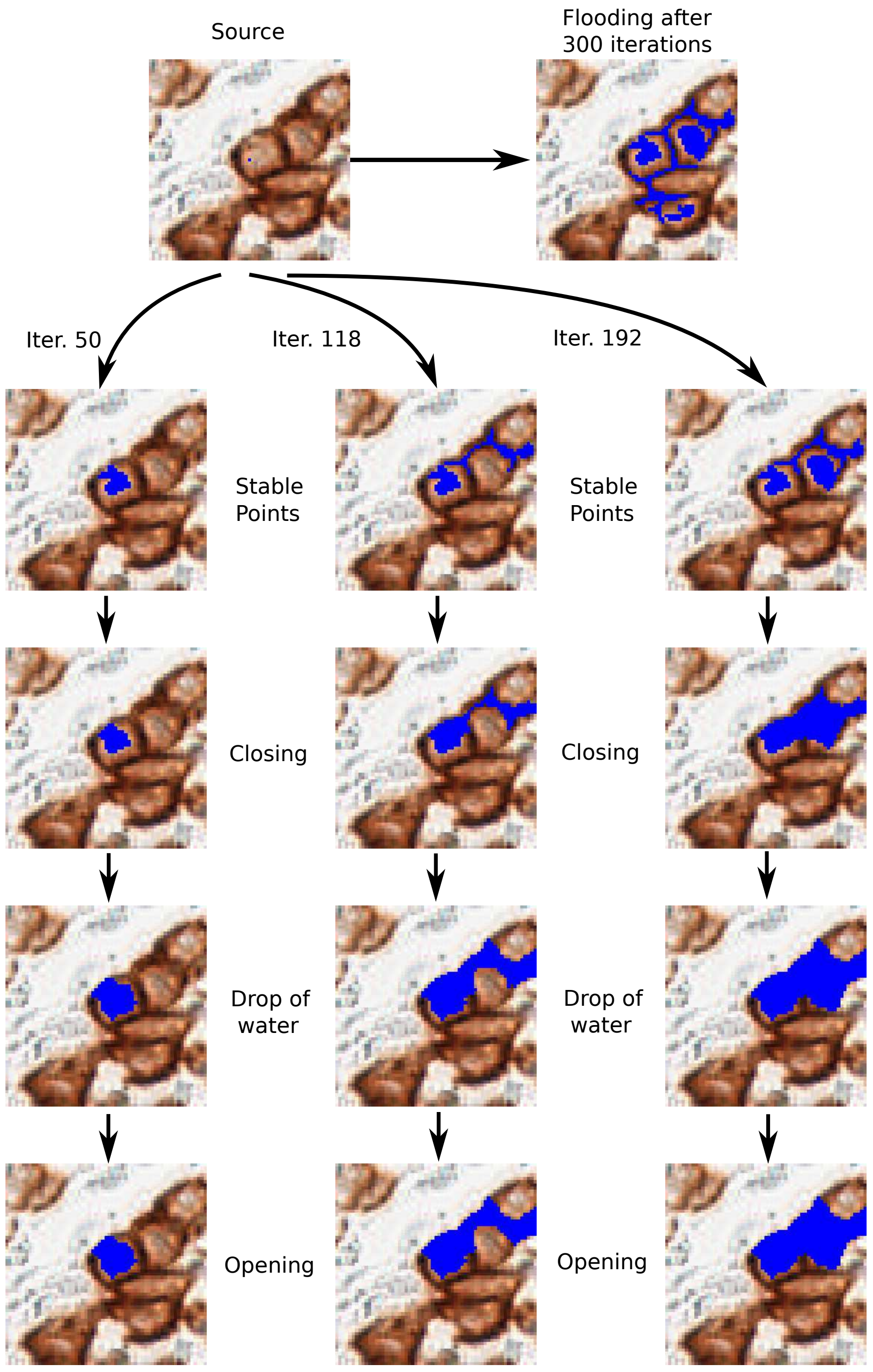


Figure 6.5: Whole process of the flooding segmentation for a cluster of cells. The initial source in the cell originates a flooding pattern (top right) of 300 pixels long (called iterations), but only 5 stable points occur at iteration 50, 118, 192, 221 & 240 (in this figure only the first 3 are analysed). The raw segmentation (second row) at the stable points are processed by applying closing (third row), drop of water principle (fourth row) and opening (fifth row).

7

Flooding Detector: Feature Extraction & Evaluation

In the *flooding detector*, hypothetical cell centres are generated using a difference-of-Gaussians detector (Chapter 4), then they are extended to segments using the flooding segmentation (Chapter 6), and finally fed into an SVM classifier (Chapter 2). This chapter will describe and evaluate the different features tested for the classifier.

In Section 7.1 the features will be described. The evaluation of the features will be addressed in Section 7.2.

7.1 Features

The features to be studied can be put into three different groups: size-dependent shape features, size-independent shape features, and colour features. The majority of these descriptors were inspired by Andersson [31].

The term '*(binary) flooding mask*' will be used in this chapter, which indicates how the segmentation is represented in the code. A binary mask was created for each point hypothesis, where '1' means that the pixel belongs to the segment, and '0' otherwise.

7.1.1 Size-Dependent Shape Features

The size-dependent shape features are:

- *Area:* It is defined as the number of foreground pixels in the flooding mask.
- *Perimeter:* It is defined as the number of foreground pixels in the flooding mask for which at least one of the eight neighbours is a background pixel.
- *Thickness:* It is defined as the number of shrinking steps (elimination of border pixels one layer per step) to make an object disappear. More precisely, for each

shrinking step the thickness is increased in 2 except in the last step, in which the thickness is increased in 1 if the remaining object has a $width = 1$, and 2 otherwise. For example, the thickness in a circle indicates the radius.

- *Convex area:* It is defined as the area of the smallest convex hull that contains the flooding segmentation.
- *Convex perimeter:* It is defined as the perimeter of the convex hull.

A convex hull of an object in the Euclidean space is the smallest convex set that contains the object. Recall that a convex set is a set such that for any two points in the set, the line between them is also contained in the set. This concept is illustrated in Figure 7.1.

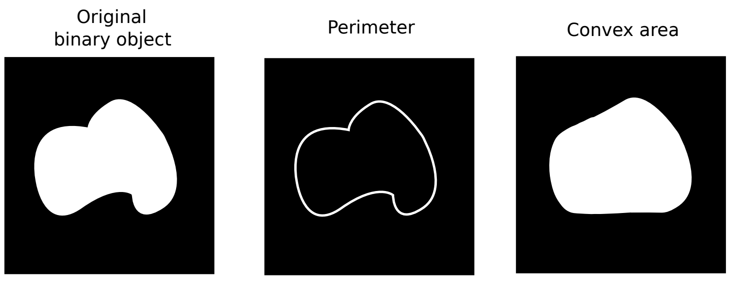


Figure 7.1: Descriptors. Pixels with value 1 are white, pixels with value 0 are black. **Left:** The area of the object is the number of white pixels. **Centre:** The perimeter of the object. **Right:** The convex area of the object.

7.1.2 Size-Independent Shape Features

The size-independent shape features are just functions of the size-dependent ones. Those are:

- *Formfactor:* It is defined as
$$4\pi\frac{\text{Area}}{\text{Perimeter}^2} \quad .$$
It is used to distinguish between round and non-round objects. The circle is the object with the highest value, 1. As the shape of an object becomes rounder and smoother, the formfactor will approach one, whereas as the shape becomes less round or less smooth, the formfactor would approach zero.
- *Elongatedness:* It is defined as
$$\frac{\text{Area}}{\text{Thickness}^2} \quad .$$
It attempts to measure how long an object is. The circle is the object with the smallest value. Elongatedness will tend to ∞ as the object becomes longer and thinner.

- *Convexity:* It is defined as
$$4\pi\frac{\text{Convex perimeter}}{\text{Perimeter}} .$$
It measures the convexity of an object. The circle is an example of an object with the highest convexity, 4π. When a shape becomes less smooth (or rougher), the perimeter of the shape might increase very quickly, whereas the convex hull perimeter changes at a much slower rate. Therefore, as the shape becomes less smooth, the convexity will approach zero.

- *Solidity:* It is defined as
$$\frac{\text{Area}}{\text{Convex area}} .$$
It measures the overall concavity of an object. The circle is an example of an object with the highest solidity, 1. As a shape becomes rougher, the solidity value will approach zero.

7.1.3 Colour Features

Two descriptors are used: **mean colour** and **standard deviation**. The RGB colourmap is chosen for this purpose. This provides six feature values.

The same features are computed again after the original segment is eroded with a diamond structure of size 4. The idea behind this new feature is to get rid of the brownish ring in the edges of the cells, leaving only the inner part. Since there can be segments of different sizes inside an image, we need to handle cases where segments were completely removed after the erosion. Three different ways of handling this were explored:

- *Total:* If an object disappears, the feature vector for that object will be zero.
- *Adaptive:* An erosion is applied with an structure of size 1 each time (4 times), and the features will be obtained from the object previous to its disappearance.
- *Partial:* If an object disappears, the erosion is not applied at all.

The method that was finally chosen was the *partial* one since it provided better results. Both descriptors, mean colour and standard deviation, were normalized based on the Euclidean distance. In total, twelve feature values are obtained.

7.2 Feature Evaluation

The training set. In order to build the SVM model, a training set of 960 positive hypotheses and 37 751 negative hypotheses was built. This training set is essentially the same used in the *rank-SVM detector*, which had 1000 positives and 7000 negatives. The initial 7000 negative hypotheses gave 37 751 different segmentations, but some of the 1000 correct hypotheses failed to generate a good segmentation. Note that for each negative point all the generated segmentations were included in the training set, whereas for the positive points only the best one was included.

The evaluation. In order to visualize how well the features separate positive and negative training examples, they were plotted in a 2D plane, two by two.

The colour features are shown in Figures 7.2, 7.3, and 7.4 for red, green, and blue, respectively. Looking at the graphs for the colour features, several conclusions can be drawn:

Note: the positive class (blue) is drawn over the negative class (red), so some negative examples can be hidden under the positive ones.

- The three mean colour features (without erosion) provide a very good separation of classes.
- Mean colour features with erosion behave slightly worse than without erosion. In order to assess if using both modalities of the mean colour feature is profitable, some tests need to be done.
- The three standard deviation features show a rather poor result, although red colour looks slightly better than green and blue. Barely any good separation could be done if only these features were used.

The size-dependent shape features are shown in Figure 7.5 for area, perimeter, convex area and convex perimeter, and in for Figure 7.6 thickness. The size-independent features are shown in Figure 7.7. All of them provide certain separation of the classes, although the overlap is noticeable. Our assumption is that large false hypotheses tend to have anomalous shapes, and thus can be easily separated from true hypotheses. However,

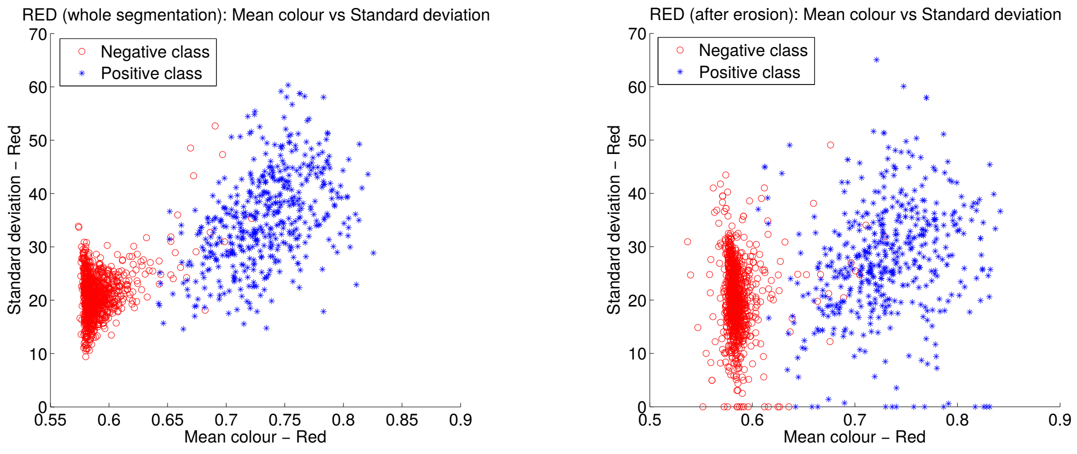


Figure 7.2: Feature space mean colour and standard deviation for RED colour. Validation set: only 500 positives & 1000 negatives, randomly selected. **Left:** Whole segmentation. **Right:** After eroding the segmentation with a disk of radius 4. **Comparison:** Mean colour separates the classes quite good in both cases. Standard deviation provides a rather poor separation, specially after the erosion where the negative class is spread out.

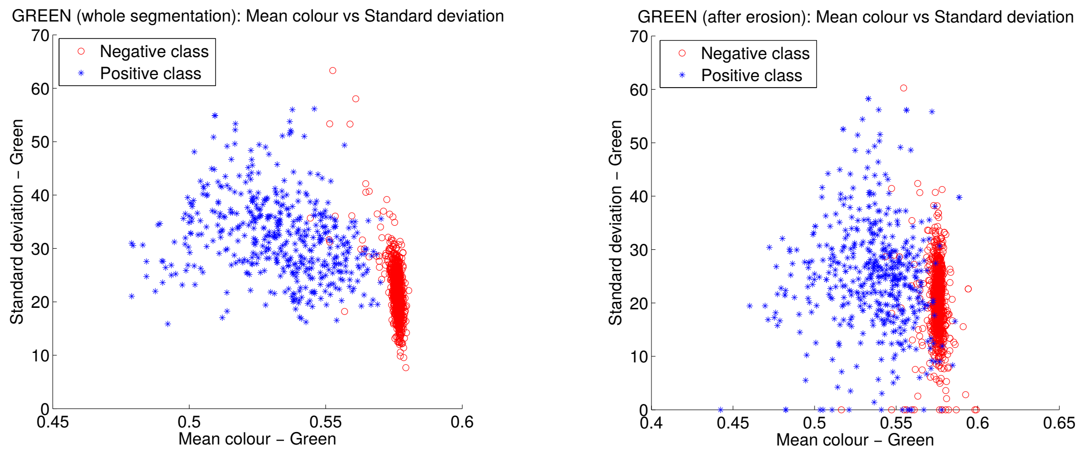


Figure 7.3: Feature space mean colour and standard deviation for GREEN colour. Validation set: only 500 positives & 1000 negatives, randomly selected. **Left:** Whole segmentation. **Right:** After eroding the segmentation with a disk of radius 4. **Comparison:** Mean colour separates the classes quite good, but the erosion degrades the performance (the classes are now closer and slightly mixed). Standard deviation provides a rather poor separation, and the erosion does worsen the separation.

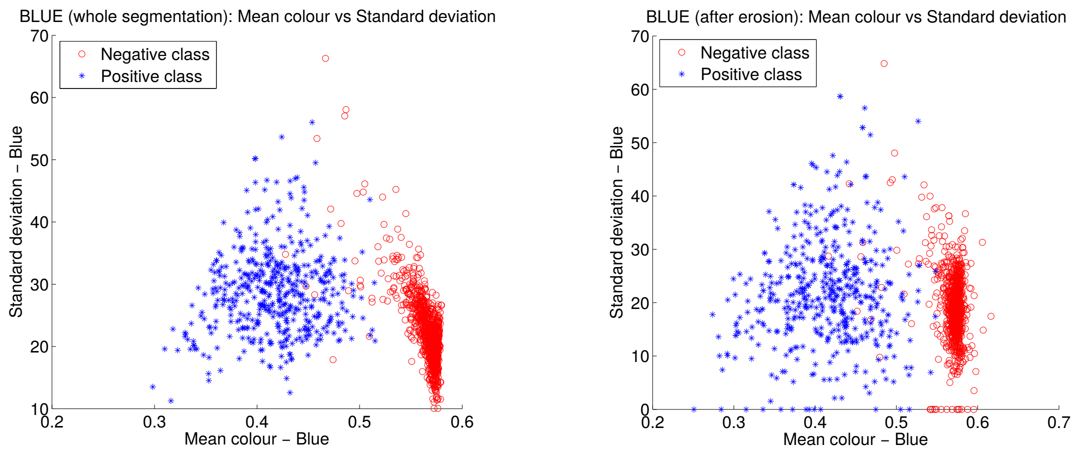


Figure 7.4: Feature space mean colour and standard deviation for BLUE colour. Validation set: only 500 positives & 1000 negatives, randomly selected. **Left:** Whole segmentation. **Right:** After eroding the segmentation with a disk of radius 4. **Comparison:** Mean colour separates the classes quite good in both cases. Standard deviation provides a rather poor separation.

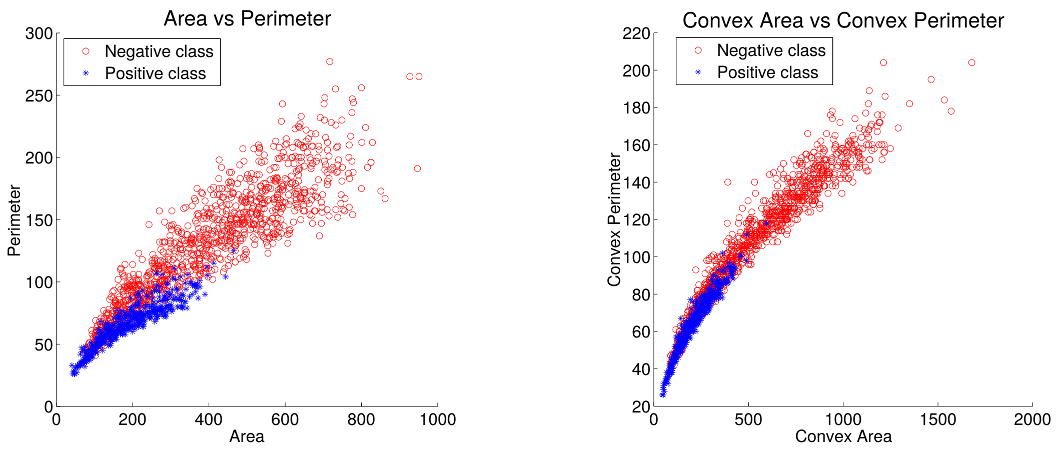


Figure 7.5: Feature space for the size-dependent features area, perimeter, convex area, and convex perimeter. Validation set: only 500 positives & 1000 negatives, randomly selected.
Left - Area vs perimeter: Noticeable overlap. Area and perimeter of the true hypotheses tend to be smaller than the average false hypothesis.
Right - Convex area vs convex perimeter: Noticeable overlap. True hypotheses tend to have a smaller convex area and convex perimeter, which means that the convex hull of the true hypotheses are close to the real segmentation. This situation happens for convex objects, which we believe our true hypotheses are.

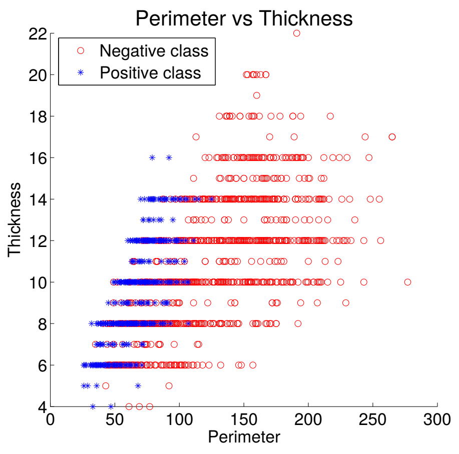


Figure 7.6: Feature space for the size-dependent feature thickness (perimeter is used for illustrative purposes). Thickness separates the classes poorly.

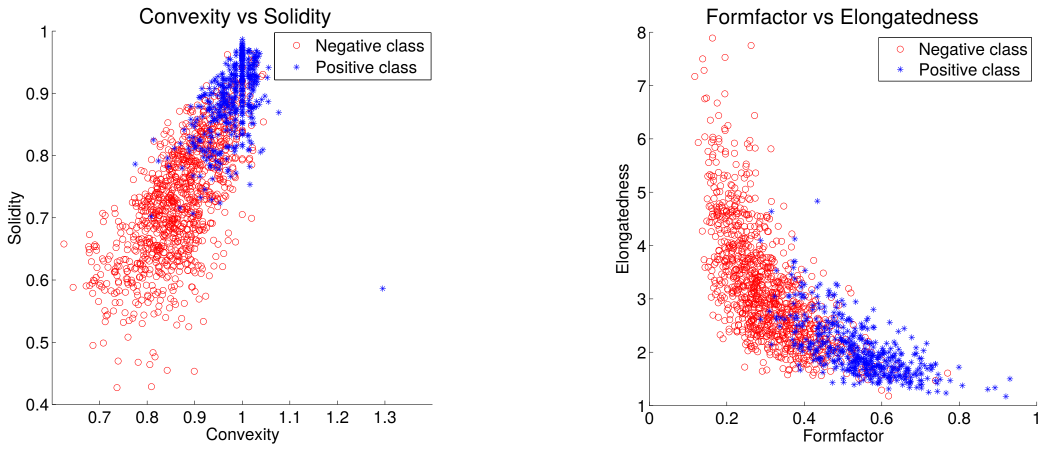


Figure 7.7: Feature space for the size-independent features convexity, solidity, formfactor, and elongatedness. Validation set: only 500 positives & 1000 negatives, randomly selected. **Left - Convexity vs solidity:** Noticeable overlap. As stated before, we believe our true hypotheses are convex objects, which means that the convex hull of the true hypotheses are close to the real segmentation. Thus, convexity and solidity tend to have a value 1 for true hypotheses (in that point, the perimeter and area of the convex hull of the object is the same as the actual object). For false hypotheses, the solidity also decreases because of the presence of holes inside the segmentation.
Right - Formfactor vs elongatedness: Noticeable overlap. Formfactor tend to be 1 for true hypotheses, which means that they are close to be circular. In this sense, false hypotheses tend to be more elongated, fact that is also registered in the feature elongatedness.

false hypotheses of correct sizes tend to also have a similar shape, and thus separation is more difficult. Some conclusions are described in the captions of the figures.

Some of these features are related between each other and provide similar information about the classes, and so it is necessary to test if the class separation in our model is improved by using all of the shape features. In order to assess this, a *backward selection* was performed. A *backward selection* involves to train first a model with all the features together and compute the error rate, and afterwards to remove one feature each time and to evaluate again the error rate. If removing one specific feature implies an improvement in sensitivity and/or specificity, then that feature should be removed. The results were (using a quadratic kernel with $C = 1$):

- The error rate for the whole feature vector is: $SE = 95.00\%$ and $SP = 99.18\%$.
- Thickness is the only feature that, after removing it, the sensitivity improves slightly while keeping the same specificity: $SE = 95.30\%$ and $SP = 99.18\%$. Thus, that feature should not be used.
- Removing any other shape feature involves decreasing the sensitivity around 0.20 - 1% while keeping the same specificity.

- The standard deviation, against the first impression, it is profitable: removing any of them involves decreasing the sensitivity around 0.5%, although the specificity improves in 0.02%. Thus, we believe that they should be included.
- Not including the 6 colour features from the eroded segmentation involves decreasing the sensitivity in 0.10% and the specificity in 0.03%. Despite the small improvement we keep those features as they do not increase the computation time.

Imbalanced training set. The training set, which has 960 positive elements and almost 38 000 negative elements, is clearly imbalanced towards the negative class. If no correction is introduced, a SVM classifier would give more weight to the specificity to the detriment of the sensitivity.

We tested several misclassification costs while training the SVM model. A misclassification cost of 1:x means that, when the hyperplane is computed, misclassifying a positive training-element costs x times more than misclassifying a negative one. Decreasing that parameter involves increasing the specificity but decreasing the sensitivity; inversely, increasing that parameter involves decreasing the specificity but increasing the sensitivity.

The misclassification cost of 1:38, which is the one that supposedly would balance the data set in weight, gives too much weight to the sensitivity to the detriment of the specificity (see Table 7.1). As shown in Figure 7.2, the 'core' of the classes are actually quite separated, so a good balance could be done only considering the elements that lie somewhere in the zone between classes. This information cannot be obtained in practice, but we can test different misclassification costs and keep the one that provides a good balance between sensitivity and specificity. After several tests, we decided to set a misclassification cost of 1:3. The results of some of those tests are shown in Table 7.1.

This problem might be also addressed by reducing the negative set so both classes have the same amount of elements. This second option usually entails decreasing the specificity too much since information is being thrown away. The last column in Table 7.1 shows the results for this case, where negative elements (randomly selected) were discarded. Nonetheless, this could be improved if the balance of the sets is done using some kind of statistical process which, for example, would remove redundant elements.

Misclassification cost	1:1	1:2	1:3	1:4	1:5	1:10	1:20	1:38	Balanced set
Sensitivity	93.30%	94.40%	95.30%	95.50%	95.50%	96.00%	96.50%	96.50%	97.20%
Specificity	99.38%	99.24%	99.18%	99.12%	99.11%	98.99%	98.79%	98.78%	98.20%
Precision	95.59%	94.68%	94.36%	94.00%	93.90%	93.20%	91.99%	91.56%	88.28%

Table 7.1: Sensitivity, specificity and precision in the validation set when training the hyperplane with a specific misclassification cost (1:x means that misclassifying a positive element costs x times more) - The last column indicates the values when the set has been balanced by selecting (randomly) the same amount of negative elements than positives

SVM kernel. It should be mentioned that all the kernels tested (linear, quadratic, cubic, and Gaussian $\beta = 1$) gave similar results. Practically all the segmentations created from false point-hypotheses (outside cells) were equally rejected by all the kernels, and the best segmentations from true point-hypotheses (inside cells) were also equally classified as positives.

There is though a small difference. As we showed in Chapter 6, sometimes a point-hypothesis inside a cell creates a segmentation covering more than one cell. The linear kernel has shown some difficulties to classify these cases as negatives, as they should be, being this misclassification two times higher than in the other kernels. Quadratic kernel is the one that removes slightly more of those cases with respect to Gaussian or cubic, so this was the selected kernel for the final design. Nonetheless, this is a minor problem since the *final selector* should also reject those cases.

Probabilistic output in the SVM classifier. Platt's probabilistic output (Chapter 2) was included in the design. This value will be used as input in the last block, the *final selector*. Note that, since several segmentations can be generated from the same point-hypothesis (and more than one can be classified as positives), it might be useful to use a probabilistic value to choose the best one among them. This problem can be understood by looking at Figure 6.4 from previous chapter: If the three segmentations are classified as positives, it would be preferred to choose the last one as the final solution. Nevertheless, we could have chosen any of the three by any other method and still the detection, our real goal, would most probably be the same.

Conclusion. In summary, all features with the exception of thickness were used, and the SVM model was built with a quadratic kernel ($C = 1$ - best value) and a misclassification cost of 1:3. This provided the following results:

$$\mathit{Sensitivity} = 95.30\% \quad - \quad \mathit{Specificity} = 99.18\% \quad - \quad \mathit{Precision} = 94.36\%$$

8

Common: Final Selection

In the *rank-SVM detector*, hypothetical cell centres are generated using a difference-of-Gaussians detector (Chapter 4), and fed into an SVM classifier (Chapter 2). In the *flooding detector*, a similar structure occurs, extending the centre-hypotheses into segments using the flooding segmentation (Chapter 6) before entering the classifier. In both cases, a final step is necessary to remove some true hypotheses that are in conflict with others because of their proximity. The core of the selector is common for both detectors: the *Maximum Weighted Independent Set* (maximum WIS) is used to solve the problem. Some small differences appear between both designs, but this does not modify the structure of the selector in its essence.

This chapter will describe the method to select the best true-classified hypotheses. In Section 8.1 the approach to solve the problem will be described, discussing the motivation behind, along with the details regarding the *rank-SVM selector*. The details for the *flooding detector* will be mentioned in Section 8.2. Finally, a description of a new evaluation method that includes the *final selector* will be elaborated in Section 8.3.

8.1 The Selection Problem in the Rank-SVM Detector

In the *rank-SVM detector*, the selection problem can be formulated as if it were a maximum WIS problem in the following way:

In an image with true classified hypotheses, each hypothesis will cover certain circular space equal to the area of an average cell (Figure 8.1-centre). If two hypotheses are too close, they are unlikely to represent different cells. Thus, we define a length, called '*conflict distance*', which is the smallest distance between two compatible hypotheses. Based on that, a graph of N nodes and M edges will be built, where N is the number of true hypotheses, M is the number of total conflicts, and a node n_a will be connected to

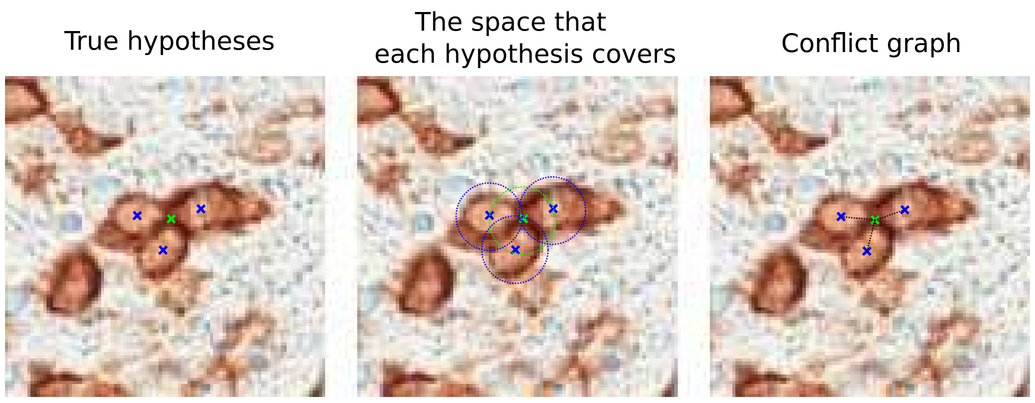


Figure 8.1: Optimization problem. The green hypothesis is the core of the problem: **Left:** The initial hypotheses inside a cluster of cells that have been classified as positives. **Centre:** The theoretical space that each hypothesis covers. **Right:** The conflict graph, wherein a line between two hypotheses indicates that they are in conflict and thus only one of them could stay.

another node n_b by an edge e if the distance between them is smaller than the conflict distance (Figure 8.1-right). Moreover, each node will have a weight equal to the output of the rank-SVM classifier for the correspondent hypothesis.

If the resulting conflict graph is disconnected, it can be subdivided in several connected graphs, so each one can be solved independently.

Why is the maximum WIS problem a good approach to solve this matter? In our problem it is expected that the 'best' true hypotheses (the ones in the centre of the cells) have a higher rank-value than the rest. However, this assumption cannot be assured for all the cases, and some exceptions might occur. Our tests showed that, normally, the closer a hypothesis is to the edge of a cell, the smaller the rank-value. However, this pattern varies for clusters of cells, and some hypotheses inside the cell might have a slightly higher score than the hypothesis in the actual cell centre. Nevertheless, those exceptions are usually in conflict not only with the hypothesis in the centre of that cell but also with hypotheses in the centre of other cells.

Thus, we believe that the maximum WIS problem can provide a good solution for the majority of the conflict graphs. The maximum WIS would select the combination of hypotheses that maximizes the sum of their rank-values, which would avoid the problem where a bad hypothesis with a high rank-value would impose upon good hypotheses with lower rank-values.

This scenario can be better understood in Figure 8.1-right. Even if the green hypothesis has a slightly higher rank-value than any of the blue hypotheses, maximum WIS solves the problem correctly, selecting the three blue hypotheses as the final solution.

Solving maximum WIS. The maximum weighted independent set problem will be solved differently depending on the size of the conflict graph. As was mentioned, the whole conflict graph from an image would most probably be disconnected; if so, that graph will be subdivided in several connected graphs so each one of them can be solved independently.

If one conflict graph is small (less than 50 nodes or 1000 edges in our implementation), an exhaustive search will be used to solve it. An exhaustive search is exponential in time complexity, so we avoid to use it if the graph is too large. For those cases, a heuristic approach is chosen where the method selects first the node with the highest weight in the graph and then removes the adjacent nodes, iterating the process until all nodes have been selected or removed. In other words, this heuristic method just employs a greedy approach, which has linear time complexity. Thus, we expect that this heuristic approach will provide the solution or an approximated solution for the maximum WIS problem for big graphs.

8.2 The Selection Problem in the Flooding Detector

In the *flooding detector*, the provided segmentations allow for a more simple approach: We know exactly the amount of overlapped pixels between two hypotheses in conflict.

In our implementation, a tolerance ratio was included to manage the overlapping. We considered that if two hypotheses overlap in only a few pixels, they could still represent different cells. That tolerance ratio was set to 10% of their pixels.

The weights in the nodes are now defined differently. We seek the combination of hypotheses that maximizes this rule:

Sum of their Platt's output (normalized) + number of pixels of the whole combination (normalized) + number of hypotheses (normalized)

This approach changes slightly how the maximum WIS solution is obtained. The normalization of those values involves that the weights of the nodes are not fixed. In other words, first all the independent sets are computed, then for each one of them those parameters are obtained, the parameters are then normalized (independently) based on the highest case, and finally they are added together.

The motivation behind choosing that rule (instead of just simply the Platt's output) involves two main issues:

- If a segmentation covering two (or more) cells is classified as positive (see Figure 6.5), we do not want it in our final solution. Possibly, the alternative of having one segmentation per cell would also be available. Then, including the rule '*maximization of the number of hypotheses*' would make our algorithm reject that case for the alternative.

- If a cell contains two different, compatible, small segmentations classified as positives (rare case, but possible), we do not want them in our final solution. Possibly, a third segmentation covering the whole cell would also be available, most probably containing a few more pixels than the previous two together. Then, including the rule '*maximization of the number of pixels*' would make our algorithm reject that case.

Finally, the implementation of the maximum WIS problem was improved computationally with respect to the problem in the *rank-SVM detector*. Since more than one source could originate the same segmentation, duplications were removed before building the maximum WIS graph, keeping the hypothesis with the most centred source. This reduced the number of hypotheses to less than 1/3.

8.3 Evaluation Method for the Final Selector

In Chapter 5 (*rank-SVM detector*) and Chapter 7 (*flooding detector*) the evaluation methods for the features were described. In both cases, a training set and/or a validation set was employed for that purpose. However, those sets were built with hypotheses classified independently. This means that the *final selector* was not included in the evaluation method.

The selector is likely to modify the sensitivity and specificity to a small extent. To be specific, the sensitivity is likely to be reduced, whereas the specificity is likely to be increased.

We seek to evaluate the output of the *final selector*, which is also the evaluation of both detectors in their entirety. Therefore, a different validation set is required. In this case, an image with 477 cells (Figure 8.2) was used as a validation image, and the number of errors of different kinds was counted manually.

An example of how the manual counting was done is shown in Figure 8.3. The meaning of the colored points is:

- **Blue**: Clear detection. One hypothesis in one cell.
- **Green**: Possible multiple detection (more than one hypothesis in one cell).
- **Red**: Unclear detection.

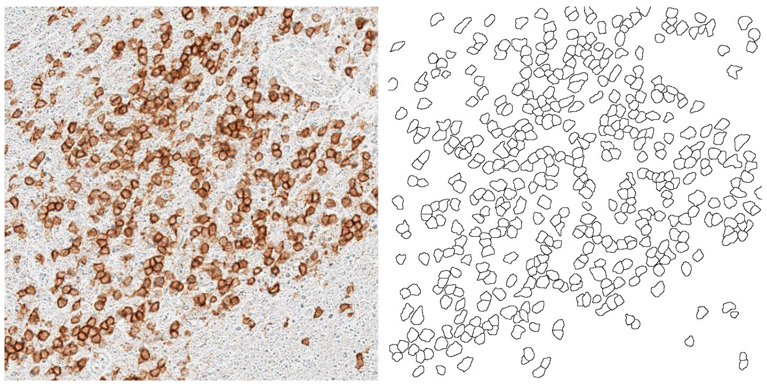


Figure 8.2: Validation image for the detectors.

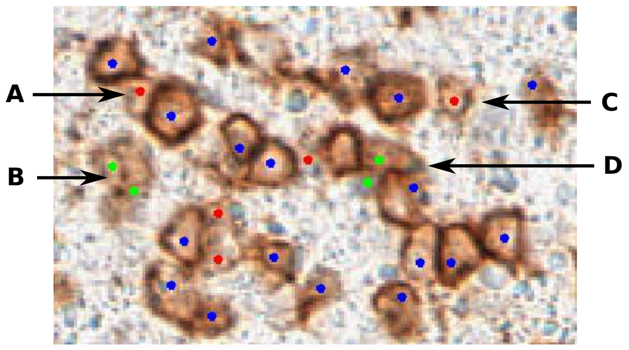


Figure 8.3: Illustration of how the manual counting was done. Each point indicates a cell detection from the *flooding detector*, and they were manually labelled: Blue indicates clear detection, red indicates unclear detection, and green indicates a possible multiple detection.

Let us discuss the image:

- Case A & C are 'weak cells', term that was introduced in Chapter 4. Those are partial cells not properly stained. Should they be considered cells or not? What is the 'limit' when a weak cell should be interpreted as a cell? The answer to those questions might vary depending who is asked, i.e. it is a subjective question even for medical experts. In this case, we considered that A & C were correctly detected. Note that the other three red points in the image are also weak cells, but in those cases we kept them as false detection.

- Case B shows a double detection in one cell. However, is that truly a single cell? Again, the interpretation of the image is quite subjective. In this case, we kept them as double detection, so this was counted as one correct detection and one error.

- Case D is also a double detection. However, a different conclusion was made here. In this case, there is a neighbour cell not detected (missed cell). Our interpretation was to consider that the second detection in D could be seen as the detection of the missed cell. A second interpretation was also discussed where the red point could be seen as the detection of that cell. In both cases, the result would be two cells correctly detected and one error.

9

Results and Discussion

In the *rank-SVM detector*, hypothetical cell centres are generated using a difference-of-Gaussians detector (Chapter 4), fed into an SVM classifier (Chapter 2), and the best combination of hypotheses positively classified are finally selected (Chapter 8). In the *flooding detector*, a similar structure occurs, extending the centre-hypotheses into segments using the flooding segmentation (Chapter 6) before entering the classifier. This chapter evaluates the final results for both algorithms, the *rank-SVM detector* in Section 9.1 and the *flooding detector* in Section 9.2.

9.1 Rank-SVM Detector

An example of the detection produced by the *rank-SVM detector* is given in Figure 9.1.

The '*conflict distance*' (term introduced in Chapter 8) was set to 12 pixels ($1.2 * radius$), which means that if two hypotheses are closer than a distance = 12, they will be in conflict. The selection of this distance is discussed below.

For this specific image, the threshold employed was the one that provided for the training set (see Chapter 5) a sensitivity of 99% and specificity of 95%.

9.1.1 Choosing the Conflict Distance

Choosing the conflict distance is non-trivial as there exist cells of different sizes. As can be seen in Figure 9.2, no choice is perfect for all cell clusters. In the final implementation a distance of 12 pixels was chosen.

9.1.2 Choosing the Threshold in the Rank-SVM Classifier

Selecting the threshold is not a straightforward decision. For instance, image contrast or intensity will affect the ranking value of the hypotheses. This problem can be illustrated by comparing the training and validation sets from the output of the classifier

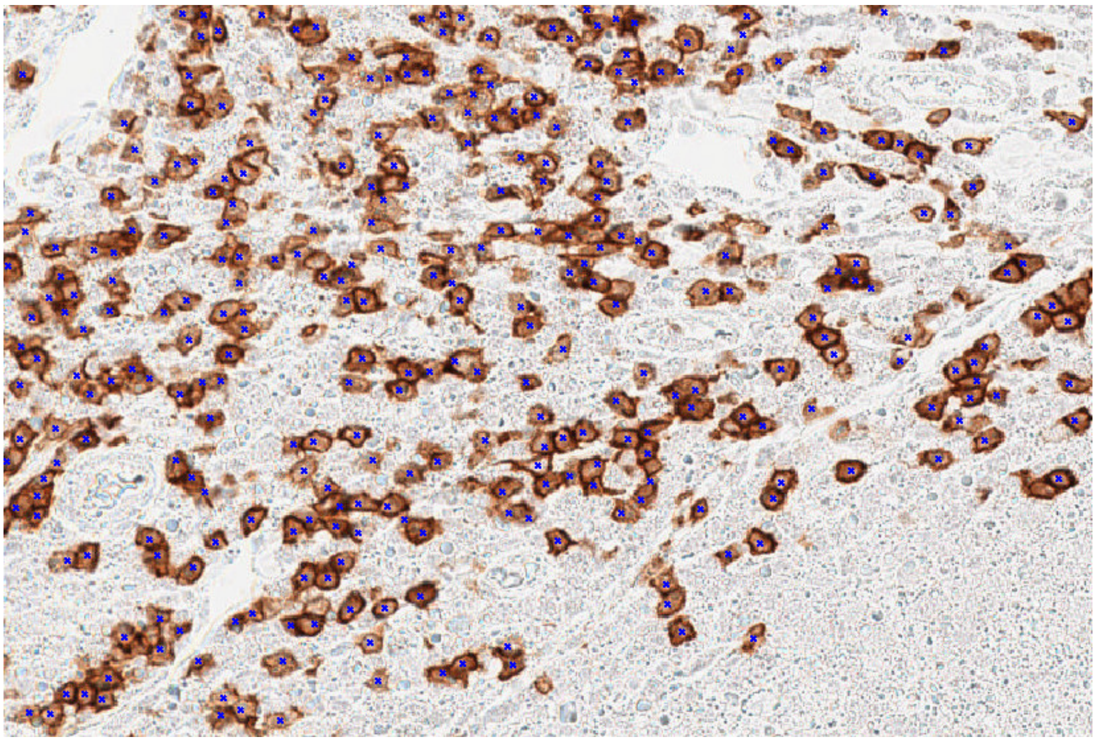


Figure 9.1: Results for the rank-SVM detector using a *conflict distance* = 12.

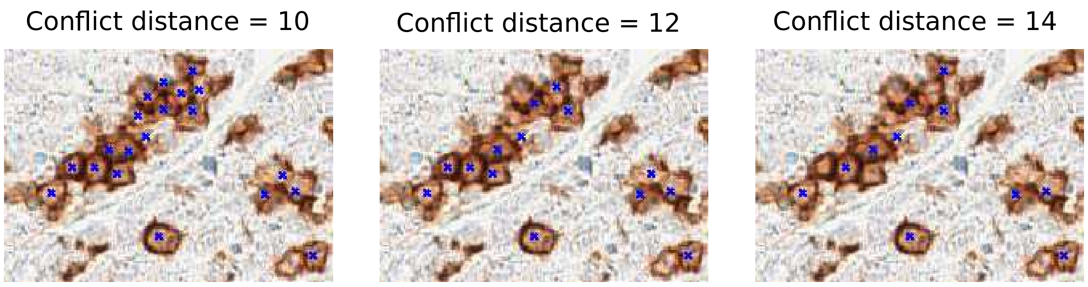


Figure 9.2: Selecting the conflict distance: **Left:** 10 px. **Centre:** 12 px. **Right:** 14 px.

(Chapter 5). This is depicted in Figure 9.3. Both sets have the same amount of positive and negative elements, and use the same rank-SVM model. However, we believe that the small difference in image intensity between both images generates the shifting in the sensitivity/specificity graph. Figure 9.3 shows that, if we move the lines from the validation set to the left exactly a distance of *threshold = 0.3*, the results of both sets will practically match. Thus, we believe that changes in contrast/intensity in the image do not affect the ranking procedure, and we can still obtain the same relation between sensitivity and specificity. However, the rank values of all the hypotheses might increase or decrease in the same proportion. In summary, we cannot choose a threshold to be employed for all images, and the user should tune it manually.

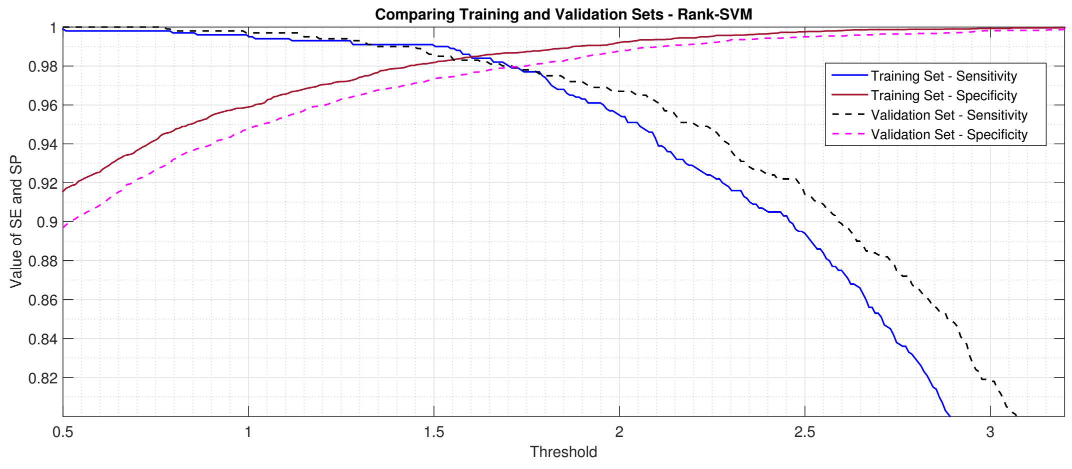


Figure 9.3: Rank-SVM classifier: Comparison between training and validation sets - Sensitivity and Specificity based on the threshold.

9.1.3 Evaluating the Detection

The validation image (see Chapter 8), which had 477 cells, is employed to evaluate the *rank-SVM detector*. In this case, only 430 of those cells will be used in the evaluation since the detector cannot work near the borders of the image. We initially set the same threshold used for the previous analysis, i.e. the one that provided for the training set a sensitivity of 99% and specificity of 95%.

The test provided these results:

Rank-SVM detector - Evaluation		
Cells in the evaluation image = 430		
Detected cells = 413	TP = 403	Sensitivity = 93.72%
	FP = 8	Precision = 98%
	FN = 27	

Table 9.1: Evaluation of the rank-SVM detector.

Note that, for the validation image (only 430 cells), 10 093 hypotheses were detected initially (output of the *raw detector*). Among those, only 866 were positively classified by the rank-SVM classifier, and the final selector considered that only 413 formed the optimal combination. In order to give a correct specificity, the initial 10 093 hypotheses should have been categorized. That task is very time-consuming, and we believe that it is safe to assume that 2/3 of those hypotheses belong to the background of the image (false hypotheses). Therefore, the specificity can be predicted to be almost 100%.

If these results are compared against the ones from the rank-SVM classifier (Figure 9.3), we can observe that the obtained sensitivity (93.72%) has provided a better specificity (from 99% to 100%). Thus, the final selector improves slightly the results, as was expected.

9.1.4 Discussion

Setting the threshold in the rank-SVM classifier is not a simple task. If the specificity is not high enough, too many false hypotheses close to the cells will tend to be classified as positives, and the final selector might not be able to remove them. On the other side, if the sensitivity is not high enough, some true hypotheses inside the cells will not be classified as positives. Both, sensitivity and specificity, cannot be perfect, so one goes to the detriment of the other. Most importantly, this decision must be done every time a image is analysed since the thresholds might change, as was explained in Section 9.1.2. In the end, the user has a necessary role in the detection. Some people might consider this an advantage of the algorithm; other might find it a drawback. The former could say that letting the user choose the threshold might empower the algorithm because he/she could make sure every time that the detection is as accurate as possible. The latter would argue then that an user might degrade the results if his/her knowledge is inadequate, or that selecting a threshold by observing images with thousands of cells might not be a trustworthy approach.

Regarding the time complexity of the algorithm, it can be considered linear if the greedy approach in the final selector is employed, which occurs for very large images.

Nevertheless, the *rank-SVM detector* might be considered a success, and its main advantage is its simplicity. By just using very simple ideas it can provide a rather good performance. The *raw detector* can be implemented in a very easy way, only two features are extracted from the image, and the *final selector* provides a good detection.

Moreover, looking at the image in Figure 9.1 is clear that the algorithm does not produce big mistakes in the detection. In fact, the spatial distribution of the final hypotheses approximates to the spatial distribution of the real cells: maybe some final true hypotheses are not in the best position, but the ‘big picture’ is rather excellent.

9.2 Flooding Detector

An example of the detection produced by the *flooding detector* is given in Figure 9.4. In this case, the tolerance ratio (term introduced in Chapter 8) was 10%, which means that two segmentations can overlap 10%.

9.2.1 Choosing the Tolerance Ratio

The aim of introducing this parameter was to avoid conflicts between hypotheses that only overlap by a few pixels. Thus, this was not supposed to be a parameter to be tuned in every analysis. Several tests showed that a tolerance around 10% is suitable.

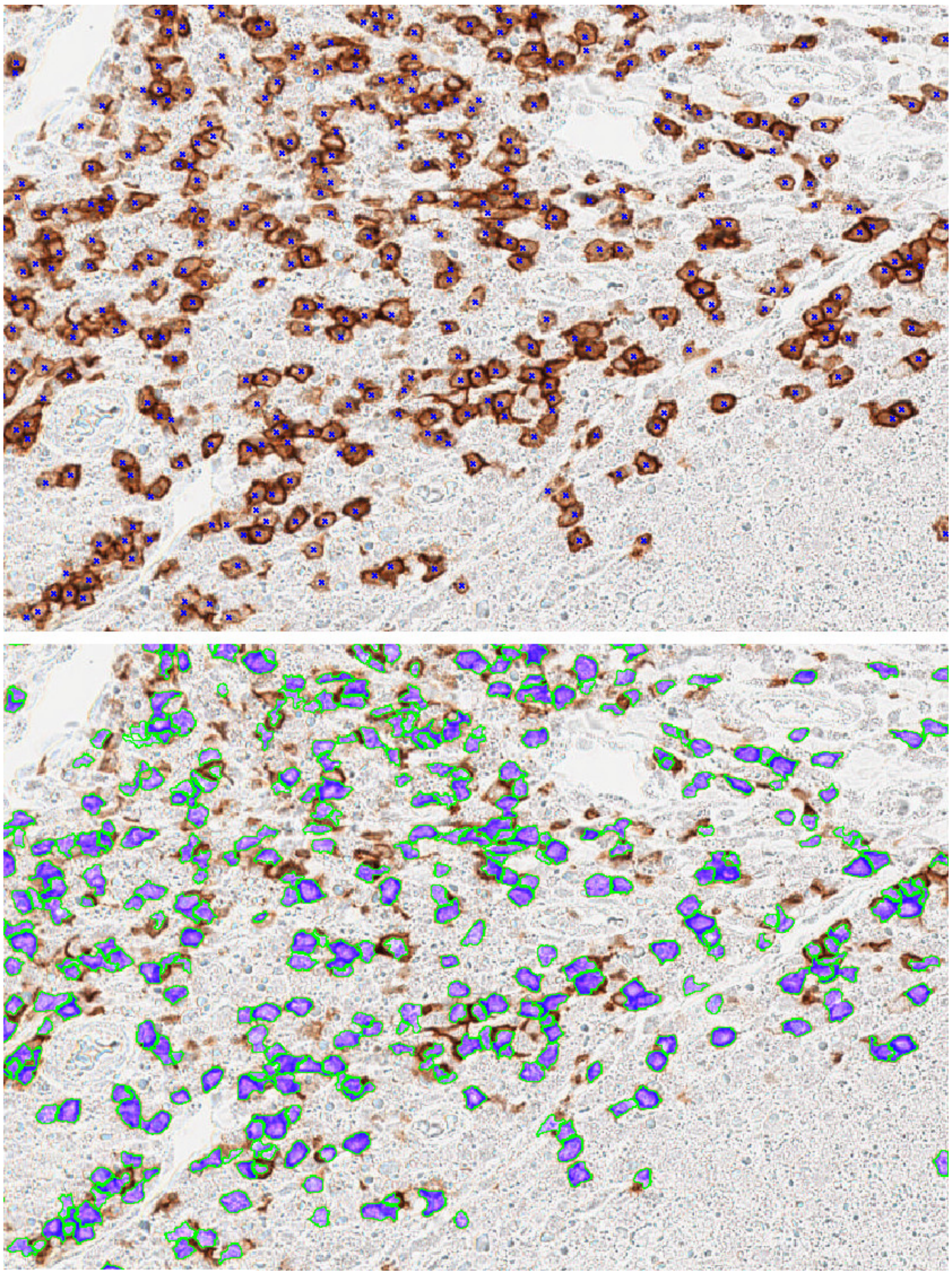


Figure 9.4: Results for the flooding algorithm. **Top** Detection. **Bottom:** Segmentation (green lines are edges, and bluish pixels are the segmentations).

9.2.2 Evaluating the Detection

The validation image, which had 477 cells, is employed to evaluate the flooding detector. The test provided these results:

Flooding detector - Evaluation		
Cells in the evaluation image = 477		
Detected cells = 522	TP = 466	Sensitivity = 97.69%
	FP = 56	Precision = 89.27%
	FN = 11	

Table 9.2: Evaluation of the flooding detector.

If we compare these results with the ones from Table 7.1, we can observe that the sensitivity has improved from a 95.30% to a 97.69%. Regarding the specificity, an assumption similar to the one made in the *rank-SVM detector* can be made:

From the validation image (only 477 cells), 11 866 hypotheses were detected initially (output of the *raw detector*). Note that this number is higher than the one in *rank-SVM detector* because the *flooding detector* can actually perform in the whole image, even in the borders. The number of total segmentations produced by those hypotheses is 42 720. Among those, only 3304 were classified as positives. Moreover, the majority of those were duplicates, so this turned out to be only 1337 positives segmentations. Finally, the final selector considered that only 522 formed the optimal combination. If we assume that 2/3 of the original hypotheses belong to the background of the image, the specificity is 99.3%, which is slightly higher than the one obtained without the *final selector* (see Chapter 7).

Therefore, the *final selector* has actually improved here both parameters, the sensitivity and specificity, with respect to the classifier, although the precision is reduced from 94.36% to 89.27%.

9.2.3 Discussion

The main problem of this algorithm involves segmentations covering more than one cell (see Figure 9.5). The reasons of this problem are primarily two, both originated in the flooding segmentation.

The first one is allowing more than one segmentation per source (this was called “selecting stable points” in the algorithm). We operated in that way so we could generate the desired segmentation for practically all the cells, although larger segmentations were also created. As was discussed at the end of Chapter 7, the SVM classifier is capable of removing a considerable amount of those large segmentations but not all of them. The *final selector* was also designed to correct that issue. However, there will be cases when it is not possible.

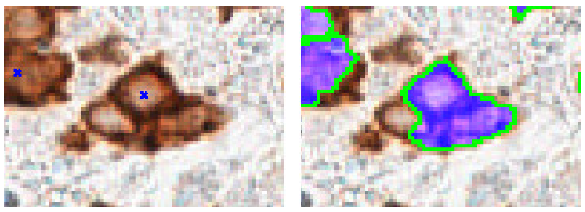


Figure 9.5: Results for the flooding algorithm with an overlapping tolerance of 10%. **Left** Detection. **Right:** Segmentation (green lines are edges, and bluish pixels are the segmentations).

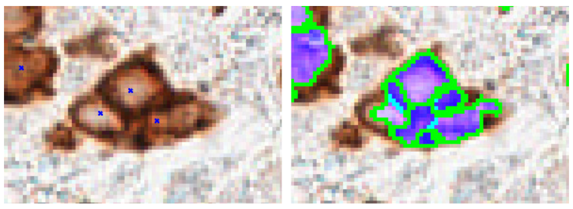


Figure 9.6: Results for the flooding algorithm with an overlapping tolerance from 55% to 100% (all produce the same results). **Left** Detection. **Right:** Segmentation (green lines are edges, and bluish pixels are the segmentations).

The second problem, exemplified in Figure 9.5, arises when the generated segmentations are not good enough. Figure 9.6 shows all the stable segmentations for this cluster. There is exactly one per cell, but unfortunately they are overlapping. The size of the overlap is 55% so we cannot solve this by a moderate increase of the tolerance ratio.

Nonetheless, besides this problem, the *flooding detector* has shown a good detection rate, and the introduction of the flooding algorithm does not involve a major complexity in the design with respect to the *rank-SVM detector*.

The time complexity of the algorithm is, in essence, the same as the one in the *rank-SVM detector*. It can be considered linear if the heuristic approach is employed, which is the case for large images.

One advantage of this algorithm is that the removal of duplicated segmentations before the *final selector* reduces drastically the size of the conflict graphs. This makes that the time employed in that part to be reduced enormously. This is one advantage with respect to the *rank-SVM detector*, where the *final selector* consumes the majority of time.

On the other hand, the flooding segmentation consumes the majority of the resources, employing around 75% of the time. Nonetheless, the code was written without taking into much consideration the time complexity; therefore, that part can be easily improved.

10

Future Work

This last chapter aims to present the lines of work for future research.

COPD Research. To be able to use these detectors to study Chronic Obstructive Pulmonary Disease, one would include images from the lung tissue and/or another type of tissue from the respiratory system. The images presented in this thesis are from the tonsils only, so it would be necessary to train detectors for other cell types and tissue.

The Rank-SVM Detector. The rank-SVM detector has proven to rank the hypotheses rather well, but the final selection often fails to identify the conflicting hypotheses. Probably a more accurate definition of conflicts (than a threshold on the distance) would be necessary. A possible solution would be to fit an ellipse to each hypothesis, yielding an approximate segmentation.

The Flooding Detector. The flooding detector has shown good results in detection and segmentation, and no detections outside the cells seem to appear. However, the algorithm is not capable of removing all the large segmentations that cover more than one cell. More work needs to be done regarding the selection of stable points. A possible solution would be to redefine the way we select those stable points, like for instance including a penalty for large segmentations.

Bibliography

[1] IHC World, “Introduction to immunohistochemistry,” 2011. [Online]. Available: http://www.ihcworld.com/introduction.htm

[2] A. Coons, H. Creech, and R. Jones, “Immunological properties of an antibody containing a fluorescent group,” *Proceedings of the Society for Experimental Biology and Medicine*, vol. 47, pp. 200–202, 1941.

[3] I. Smal, M. Loog, W. Niessen, and E. Meijering, “Quantitative comparison of spot detection methods in fluorescence microscopy,” *IEEE Transactions on Medical Imaging*, vol. 29, no. 2, pp. 282–301, February 2010.

[4] P. Nakane and G. Pierce, “Enzyme-labeled antibodies for the light and electron microscopy localization of tissue antigens.” *The Journal of Cell Biology*, vol. 33, no. 2, pp. 307–318, 1967.

[5] D. Mason and R. Sammons, “Alkaline phosphatase and peroxidase for double immunoenzymatic labelling of cellular constituents,” *Journal of Clinical Pathology*, vol. 31, no. 5, pp. 454–460, 1978.

[6] National Heart, Lung and Blood Institute, “What Is COPD?” U.S. Department of Health and Human Services, July 2013. [Online]. Available: http://www.nhlbi.nih.gov/health/health-topics/topics/copd

[7] World Health Organization, “The top 10 causes of death,” May 2014. [Online]. Available: http://who.int/mediacentre/factsheets/fs310/en/

[8] Jorgen Vestbo, et al., “Global strategy for the diagnosis, management, and prevention of Chronic Obstructive Pulmonary Disease. Global initiative for Chronic Obstructive Lung Disease,” 2013. [Online]. Available: http://www.goldcopd.org/uploads/users/files/GOLD_Report_2013_Feb20.pdf

[9] M. W. Davidson, “Molecular expressions microscopy Primer: Digital Image Processing - Difference of Gaussians edge enhancement algorithm,” March 2014, Olympus

America Inc. and Florida State University. [Online]. Available: http://micro.magnet.fsu.edu/primer/java/digitalimaging/processing/diffgaussians/index.html

[10] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed. Upper Saddle River, New Jersey (USA): Prentice Hall, August 2007, ISBN-13: 978-0131687288.

[11] H. Kong, H. C. Akakin, and S. E. Sarma, "A generalized Laplacian of Gaussian gilter for blob detection and its applications," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1719 – 1733, December 2013.

[12] T. Lindeberg, "Scale invariant feature transform," *Scholarpedia*, vol. 7, no. 5, pp. 10 491 –, 2012, category: Computer Science Computer Vision and Robotics (Autonomous Systems).

[13] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, January 2004.

[14] K. E. Atkinson, *An Introduction to Numerical Analysis*, 2nd ed. Wiley, January 1989, iSBN-13: 978-0471624899.

[15] S. Beucher and C. Lantueljoul, "Use of watersheds in contour detection," *International Workshop in Image Processing*, September 1979, rennes, France.

[16] F. Meyer, "Integrals, gradients and watershed lines," *Mathematical Morphology and its applications to Signal Processing*, no. 70 - 75, May 1993, symposium held in Barcelona.

[17] L. Najman and M. Schmitt, "Definition and some properties of the watershed of a continuous function," *Mathematical Morphology and its applications to Signal Processing*, pp. 75 – 81, May 1993, symposium held in Barcelona.

[18] F. Meyer, "Topographic distance and watershed lines," *Signal Processing*, vol. 38, no. 1, pp. 113 – 125, July 1994.

[19] J. Cousty, G. Bertrand, L. Najman, and M. Couprie, "Watershed cuts: Thinnings, shortest path forests, and topological watersheds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 5, pp. 925 – 939, May 2010.

[20] J. Angulo and D. Jeulin, "Stochastic watershed segmentation," *Proceedings of the 8th International Symposium on Mathematical Morphology*, vol. 1, pp. 265 – 276, October 2007.

[21] M. G. Uzunbas, C. Chen, and D. Metaxas, "Optree: A learning-based adaptive watershed algorithm for neuron segmentation," *Medical Image Computing and Computer-Assisted Intervention - MICCAI*, vol. 8673, pp. 97 – 105, 2014.

[22] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[23] V. Vapnik, *The Nature of Statistical Learning Theory*, 2nd ed., ser. Information Science and Statistics. Springer-Verlag New York, Inc., 175 Fifth Avenue, New York, USA: Springer, November 1999, iSBN-13: 978-0387987804.

[24] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, 1st ed. The Pitt Building, Trumpington Street, Cambridge, United Kingdom: Cambridge University Press, March 2000, iSBN-13: 978-0521780193.

[25] J. C. Platt, "Probabilistic outputs for Support Vector Machines and comparisons to regularized likelihood methods," *Advances in Large Margin Classifiers*, pp. 61 – 74, 1999, publisher: MIT Press.

[26] R. Herbrich, T. Graepel, and K. Obermayer, "Large margin rank boundaries for ordinal regression," *Advances in Large Margin Classifiers*, vol. 7, pp. 115 – 132, January 2000, publisher: MIT Press.

[27] T. Joachims, "Optimizing search engines using clickthrough data," *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*, pp. 133 – 142, 2003.

[28] O. Chapelle and S. S. Keerthi, "Efficient algorithms for ranking with SVMs," *Information Retrieval*, vol. 13, no. 3, pp. 201 – 215, September 2009.

[29] T. Fawcett, "ROC graphs: notes and practical considerations for data mining researchers," *Intelligent Enterprise Technologies Laboratory*, January 2003, hP Laboratories Palo Alto.

[30] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *Computer Vision and Pattern Recognition*, vol. 1, pp. 886 – 893, June 2005, iEEE Computer Society Conference.

[31] M. Andersson, "Weed and crop classification by automated digital image processing," Master's thesis, Chalmers University of Technology, February 1998.

[32] J. Kleinberg and E. Tardos, *Algorithm Design*, 1st ed. Addison-Wesley, March 2005, iSBN-13: 978-0321295354.