Université de Namur
Faculté d'informatique
Année académique 2013–2014

Open Multi-Approach Software Tools for Prostate Carcinoma Prognosis and Diagnosis

Anthony Karali

Maître de stage :	Ph.D. Miguel Garcia Torres, Universidad Pablo de Olavide
Promoteur :	____________________ (Signature pour approbation du dépôt - REE art. 40) Dr. Wim Vanhoof, Université de Namur

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Acknowledgement

I would like to begin by expressing how working on this project was really enthralling and this has been the case for several reasons. This project is a genuine interdisciplinary enterprise and I think this kind of initiatives are the ones having the most promising potential to address tomorrow's various challenges in a constantly evolving global society. It has been a real pleasure to work at the crossroads of multiple expert domains such as image analysis, artificial intelligence and medicine. I thank all the people that have been involved in this project and gave me the opportunity to push my research skills as well as my scientific knowledge to a whole new level.

First, I couldn't say how much I am grateful to Pr. Dr. Miguel Garcia Torres who supervised my internship and lead me through the whole project by giving me pointers, references and explanations when needed. I could also benefit from his own initial work on this project and from his valuable knowledge of the domain.

Second, I would like to express my gratitude to my promoter in my home university, Dr. Wim Vanhoof, for the time he took to supervise this master thesis and for his advice and expertise.

Third, this master internship wouldn't be what it is now without the collaboration of Pr. Dr. Alcides Chaux and his medical expertise. His help was really crucial at the end of this project and couldn't have been finished without it.

My sincere gratitude also goes to the academic and technical staff from the University of Namur for their contribution to the computer scientist I have become through the years and the support they have provided for these five years as a student.

Finally, I'd like to thank my colleagues, my family and, last but not least, my wonderful girlfriend who supported me through the thick and thin.

Abstract

At a time where bio-engineering is sufficiently capable of better and more reliable results, oncologists (and pathologists in general) still resort to classic biopsy examinations to build the prognosis or the diagnosis of their patients. These diagnostic assessments lie on a rather subjective observation of cell structure and distribution whereas computational and automated techniques can actually be used since they provide a more objective diagnosis with a quantitative approach and a rigorous training based on previously diagnosed biomedical images. In this work, we aim to develop precise, generic and automated decision making software tools for prostate carcinoma prognosis and diagnosis. We study how image analysis tools and techniques can be used for cell segmentation and cell analysis. We develop our own image processing pipelines with a multi-approach based view and we test them on a set of already diagnosed images. From there, we design tools for learning and decision aimed to help the oncologists in their work of object recognition. The aims of this project are to supply the oncologists with powerful and integrated tools they can actually use in their diagnosis workflow, to provide an open-source basis for further development, refinement and improvement and to centralise inputs for this area of knowledge.

Keywords

Decision Making Software, Digital Image Processing, Feature Extraction, Data Mining, Unsupervised Learning, Supervised Learning, Artificial Intelligence Techniques, Prostate Carcinoma

1. Introduction

This research project for automated diagnosis of prostate carcinoma (particular type of cancer) is a new initiative to expand knowledge about this field and to provide a reference frame for similar research projects.

Indeed automated diagnosis engine studies are numerous and are motivated by the same urge of dealing with cancer diagnosis by making use of computational methods instead of relying on more traditional and not always reliable diagnostic protocols but also the sake of performance and speed in the diagnostic process. More than that this kind of initiatives can also benefits to less developed areas where the expertise in particular fields such as medicine is currently lacking.

Despite those shared similarities our project aims to provide a rigorous trunk of common knowledge about the field of automated diagnosis engines from which some branches will be drawn. We also aim to provide an open-source basis for further development by making use of well-proven algorithms and well-tried tools or platforms. In the proceedings of our project we often find it sad that almost no worldwide collaborative projects have yet arisen despite the dramatic technical simplicity of doing so nowadays.

Although we could find some histology images database after long research we find it really hard to get more than a few dozens of really useful (already diagnosed) images and, also, we couldn't find only but a few studies regrouping transdisciplinary knowledge for automated diagnostic engines. This kind of barriers can really be a brake for the development of project like ours and other similar initiatives.

This is clearly with these ideas in mind that it was decided to make a first attempt at the construction of a reference open framework for knowledge for this field of automated diagnosis. This master thesis is thus the first step of a very ambitious project. This has been mostly a one person project consisting of 4 internship months at the university Pablo de Olavide in Sevilla, Spain, and a work of roughly 4 other months during which disperse amount of time were dedicated to the writing of this document. Most of our work rely on personal research but we could still rely on the previous input of Ph.D. Alcides Chaux as well as Ph.D. Miguel Garcia Torres.

The current document is divided as follows: section 2 draws an overview of automated cancer prognosis and diagnosis with both motivational and design concerns. In section 3 we reference past and current techniques of image processing applied to the field of automated diagnosis engines and we also provide a number of custom processing pipelines as the first tools of an open-source platform for further development. Section 4 goes over feature extraction and selection once the image processing stage of computing is over, with a collection of state-of-the-art methods and custom tools as well. Finally, section 5 describes the machine learning input to automated diagnosis and the processes we implemented based on a popular and open machine learning platform.

2. Overview of automated prostate cancer prognosis and diagnosis

A. Motivation

According to the World Health Organization, in 2012, cancer is still among the leading causes of death worldwide with over 14 million of cases and over 8 million registered deaths ([1]). Prostate cancer is the sixth leading cause of death for men worldwide ([2]) but even the first in the UK and the second in the US ([3]). Moreover our global society, especially in more developed countries, is constantly aging and the aging factor is critical for cancer outbreak ([4]). One could then feel the urge to have efficient, objective and fast tools in order to be able to prognose and diagnose cancer instead of relying on traditional and rather subjective approaches.

This issue has already been addressed by many other research teams ([5], [6], [7], [8], [9], [10], [11]) around the world but our purpose here was, at first, to focus on one particular cancer, which is prostate cancer, and to develop integrated and multi-approach decision making software that would be able to work on large datasets of prostate images and give consistent and valid results for a wide range of various images.

B. Design

The activity of automated cancer diagnosis is aimed to offer a powerful and reliable alternative to the classic diagnostic approach. It relies mainly on biomedical image processing, artificial intelligence techniques and statistical tools. The overview of the whole process of automated computational diagnosis in [8] gives a rather well-defined idea of which processes are involved in automated cancer diagnosis.

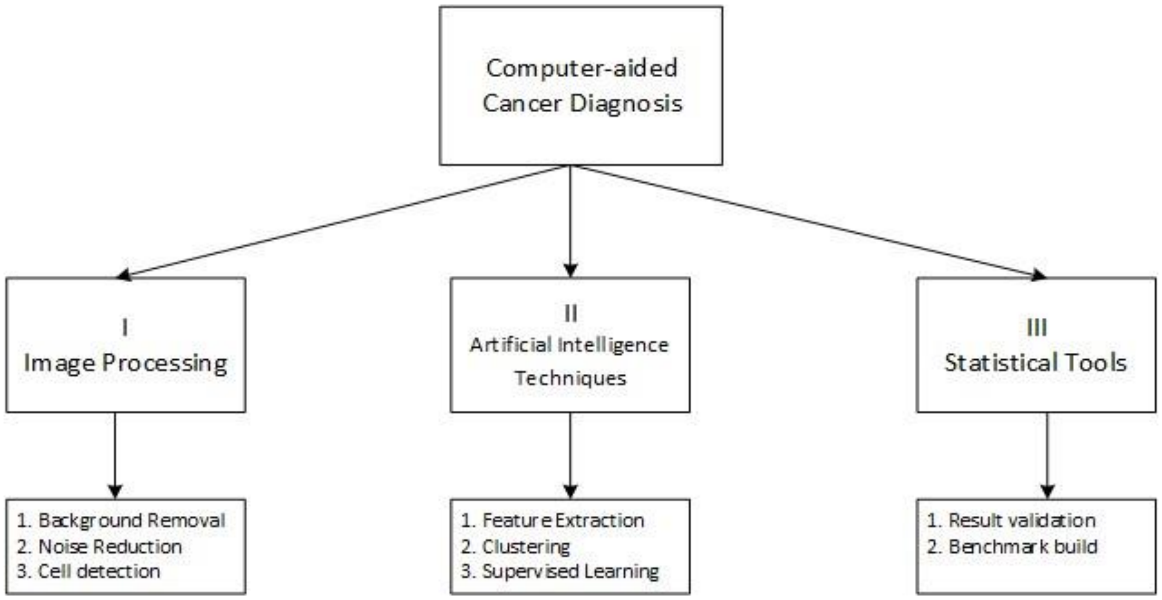


Figure 1 - Overview of the automated cancer diagnosis process

Figure 1 graphically shows how we can describe how an automated, or computer-aided, cancer diagnosis tool can be built and trained with just atomic processes combined in a defined order. Here’s a rather simple description for each of them, the next chapters and sections will go through more details:

I. **Image Processing:** a computer-aided cancer diagnosis tool is based on the analysis of biomedical images of tissue sample. We want to be able to diagnose those with decision making software (DMS) which will start its computational workflow by processing those images to further extract some features from them. The image processing step is an iterative process which will repeatedly analyse and transform each image of the study set. The study set only contains already diagnosed images for which the diagnosis has been confirmed before the insertion of the image in the study set.

   1. *Background Removal:* the biomedical images we have to work on generally contain lots of noise. A classic image processing module will start by removing as much background as possible in order to be able to further work with objects of interest only.

   2. Noise *Reduction*: once the background has been removed, it's common to perform an even more refined noise reduction to eliminate small or inadequately shaped objects.

   3. *Cell Segmentation:* the most difficult part of the image processing step is to be able to detect at least a satisfactory amount of cells, image transformation and analysis are then used to achieve this goal.

II. **Artificial Intelligence Techniques:** once we have analysed and transformed each image to have nothing but a few objects of interest left, we can start to build some knowledge about them. That's what artificial intelligence techniques are used for here. In the training process of the DMS, we don't know anything about what the detected objects really are. Thus, we have to apply some unsupervised learning techniques (here, we are going to talk about clustering only, since this is the only technique we have found to be relevant in the learning process) to classify the detected objects and then asked for an expert analysis by pathologists. Once pathologists have given some feedback about the nature of the objects, we can start optimizing the cell segmentation process and then building a supervised learning process which will allow to perform a quantitative analysis and serve as an objective diagnostic engine.

1. *Feature extraction and selection:* this step is at the border of image processing and artificial intelligence techniques. The authors of [12] defined the goal of feature selection as three-fold: "improving the prediction performance of the predictors, providing faster and more cost-effective predictors, and providing a better understanding of the underlying process that generated the data." The basic idea here is that feature selection is used to be able to manipulate only a few relevant variables, information on the characteristics of the cells, instead of an array of pixels rather hard to handle for further learning, this is a particular kind of dimensionality reduction.

2. *Clustering:* clustering is one special approach to unsupervised learning. Unsupervised learning is defined in [13] (and adjusted in [14]) as the study of "how systems can learn to represent particular input patterns in a way that reflects the statistical structure of the overall collection of input patterns". This kind of computational learning is also well contrasted in [14] with other computational learnings: "By contrast with SUPERVISED LEARNING or REINFORCEMENT LEARNING, there are no explicit target outputs or environmental evaluations associated with each input; rather the unsupervised learner brings to bear prior biases as to what aspects of the structure of the input should be captured in the output". The goal of using unsupervised learning here was to help the pathologists in their evaluation and labelling of the detecting objects after the image processing step. It's quite funny to observe that one of the earliest applications of supervised learning was used as a means of information compression in images in order to reduce the used bandwidth for image transmission ([13]). That's really the same idea with the selection of relevant features to consider for further

classifying of prostate images. The clustering step will ease the work of the pathologists by grouping the detected objects on basis of the extracted features in the previous atomic process.

3. *Supervised Learning:* Once the labelling results and other knowledge artefacts from the pathologists are received, supervised learning techniques allow to build a diagnostic engine which is trained based on what are the expected output (benign or malignant cells and, further, benign or malignant image) for given inputs (the values for each extracted feature). Indeed supervised learning can be considered as "the machine learning task of inferring a function from labelled training data" ([15]). The idea is then to use this trained engine for test images.

III. **Statistical Tools:** at every step of the diagnostic engine refinement, we have to make it sure that the results it gives are valid. For this purpose we are using some statistical tools to validate the performance of this engine. Then only, we can try to establish an image analysis benchmark for further image processing and especially, cell detection.

1. *Result Validation:* a diagnostic engine can't be used if it hasn't been tested first. This is why we have to validate its results. This can be achieved with many ways, test datasets and cross-validation techniques are however often used for this purpose.

2. *Benchmark Build:* once we have achieved good performance for the engine, using the statistical measures for the extracted features allow building a benchmark that will be helpful for further study about the cancer cell detection and cancer diagnosis.

3. Image processing

Image processing is at the core of every automated cancer diagnosis tool. It's the first and probably most critical step for the design of such device. In section 1, we review a wide range of various image processing techniques applied to the field of automated diagnosis and in section 2, we introduce our own image processing pipelines for the Prostate Carcinoma Decision Making Software (PC-DMS).

A. State of the art

At the very core of every automated diagnosis process stands a digital image processing engine whose main goal is to collect focal areas (areas where points share the same characteristics as the surroundings) from raw histological images (images taken from a microscope observation) in order to perform feature extraction and then the actual automated diagnosis. This pre-processing engine is the most critical and yet difficult part of an automated diagnosis process as there are plenty of image analysis techniques but they can be hard to combine or apply at each and every set of input images.

The raw histological images are generally produced through a staining process which allows human vision to focus on objects of interest by accurately contrasting the images.

a) Staining process: the H&E protocol

One of the oldest and widely used protocols of histologic staining is the Haematoxylin and Eosin (H&E) staining protocol. According to [16], the protocol has been used since 1904 and it's now the primary diagnostic technique in the histopathology laboratory. Basically the protocol uses the two most common staining techniques used in histology: haematoxylin staining and eosin staining. While haematoxylin best stains cell nuclei (in blue), eosin stains extracellular substances (e.g. tissue) and cytoplasm (with shades of red).

In this work, the input images were stained with this protocol and image processing techniques will be tuned to benefit from the resulting contrast between cell nuclei and tissue (or other extracellular artefacts).

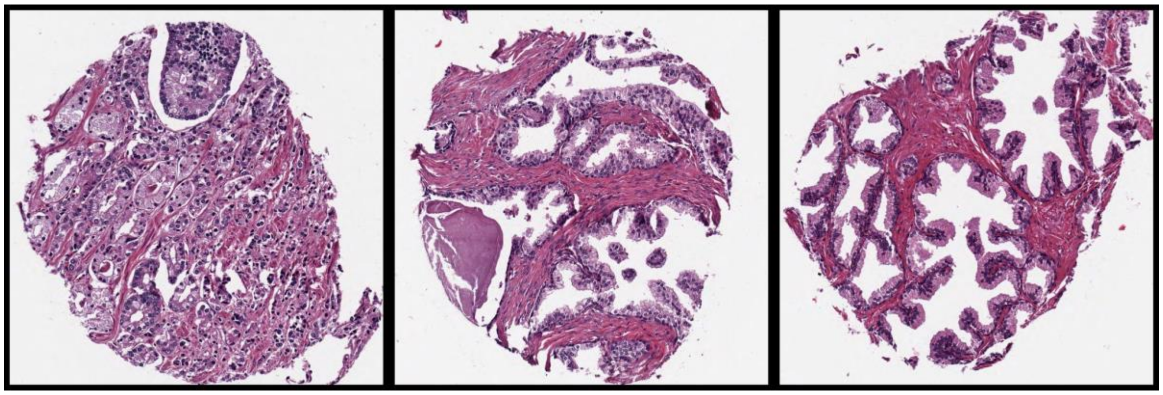


Figure 2 - Prostate images (carcinoma, normal, normal) stained with the H&E staining protocol

Now that we have seen how we can highlight the objects of interest in the input images.

Let's briefly introduce how a cancer can be diagnosed from those.

b) Cancer diagnosis: cell structure and cell distribution

There are basically two ways of using histopathological images in order to give a cancer diagnosis (automated or not) based on the properties of those. On one hand, we can study how cells are structured within the image: how they are shaped, sized, coloured, etc. We then talk about cellular-level of observation. On the other hand, we can study how cells are distributed within the image: do they form any groups? Do they seem to be randomly distributed? We then talk about tissue-level of observation. This distinction will be helpful for the following discussions as different image analysis and processing techniques can be used for each level of observation.

The most used by specialists and yet still rather subjective method for prostate cancer prognosis and diagnosis is the Gleason grading system. This method lies on the study of the two levels introduced before.

c) Cancer diagnosis: Gleason grading

In the United States, the National Comprehensive Cancer Network is responsible for the publication of several guidelines about the diagnosis and treatment of various cancers. The organization makes some assumptions about how the diagnosis of prostate cancer should be made: First, it was assumed that all patients were suspected of having prostate cancer by either an abnormal digital rectal examination (DRE) or an elevated prostate-specific antigen (PSA) determination. Second, it was assumed that the presumptive diagnosis is confirmed by a transrectal ultrasound (TRUS)-guided needle biopsy of the prostate. Finally, it was assumed that a Gleason score is assigned to the biopsy specimen by a pathologist. ([17]) The Gleason grading system is basically a scale where the higher the score is, the more advanced the cancer is. This score is obtained by analysing the biopsy images as seen previously. As introduced at the very beginning of this work, this analysis is mainly performed manually by medical experts whereas the purpose of this work and many other studies is to relate on more automatic and reliable tools, that is to say, image processing methods and artificial intelligence procedures.

In order to collect the various focal areas in the histological images and then study the histopathological properties of those, it is often necessary to partially remove the noise resulting from the staining process and more basically from rawness of the images. That’s what the next paragraph is about.

d) Image noise reduction

Noise reduction can be performed with different goals in mind. We could need a rather rough noise reduction since we are only interested in detecting coarse locations of cells or we could want to locate the cells and study them very accurately and in this case, we would need a more efficient noise reduction process. Several studies have used both methods: automated diagnosis engines requiring graph manipulations or other forms of topology analysis generally uses the former method ([18], [19], [20], [21], [22], [23]) whereas engines working with precise characterization of cells for further structural analysis would often require a more precise noise filtering process ([5], [7], [24]).

Most of the time, noise reduction processes are implemented on basis of hard- or soft thresholding techniques ([25], [26]) following some background removal or standardization step. The idea is to work only on foreground objects and then remove sets of pixels whose characteristic feature values are above or under a computed value limit set as the threshold. The critical part of such processes is then the pre-filtering phase performed before the thresholding operations as it can remove random noise. Common pre-filters are image smoothing, image contrasting or image eroding. As implemented in one of our pipelines (sequences of processes), noise reduction can also be performed without any specific thresholding techniques: it can be produced in the process flow of a colour segmentation processing pipeline. This can be achieved thanks to the colour properties of stained histological images.

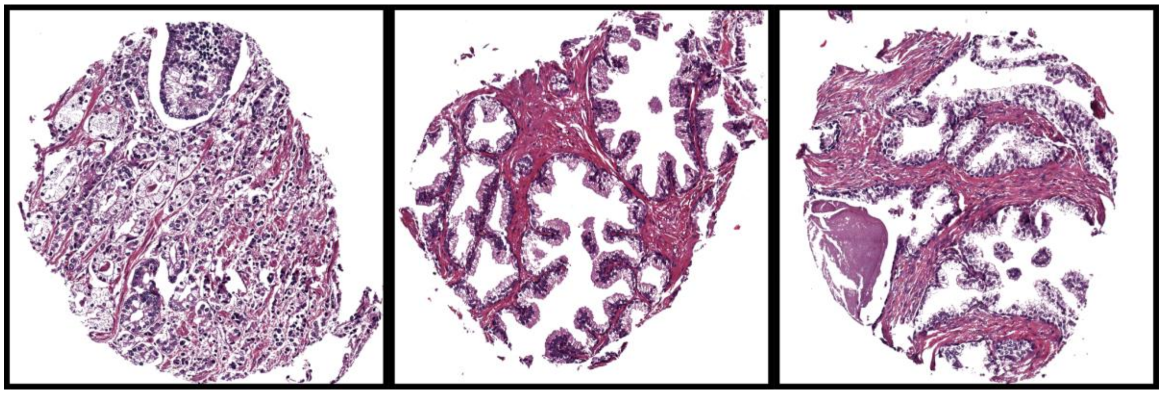


Figure 3 - H&E prostate images processed with background removal filter

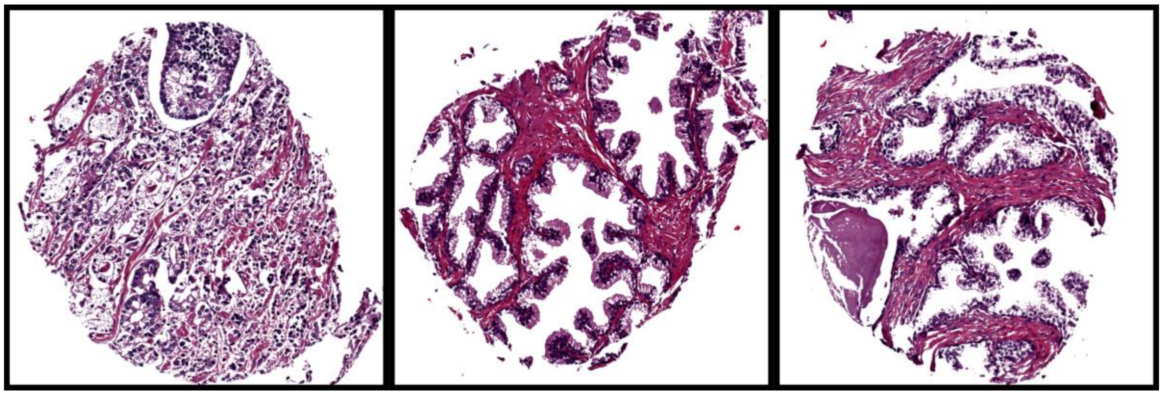


Figure 4 - H&E prostate images processed with noise reduction filter

Once a reasonably large amount of noise has been removed, image processing engines generally move towards cell segmentation (detection). This step consists of identifying objects in the images that are presumably cells. We talk about the different approaches to cell segmentation in the following discussion.

e) Cell detection

This discussion is partly based on the work of Cigdem Demir and Bulent Yener in their systemic survey of automated cancer diagnosis ([8]) and the work of Erik Meijering in his history of cell segmentation over the last 50 years ([27]).

Cell detection is among the most critical processes in an automated cancer diagnosis system. The way it can be performed will depend on how the cells are to be used in the next process, which is cell feature extraction. As mentioned, structure (or morphology) and topology information can be used in order to produce an assessment of the cancer stage in one histological image. For the former method, real precise information about the location, shape, colour and texture properties is needed. As for the latter, we would only need coarse but still relevant information about the cell properties.

Moreover, one critical issue that will affect any cell segmentation process is the variability of histological images. Indeed these images can be captured and processed with so many different devices, techniques or specifications that the first challenge for the cell segmentation process implementation is often the choice of the most relevant image processing techniques to use in order to get significantly good results. From these two different goals and the mentioned constraints arise two families of cell segmentation approaches which rely on different image processing techniques: the region-based approaches and the contour-based approaches.

We are using the same notation as in [8].

i. *Region-based approaches*

- **Thresholding:** one of the oldest and still mostly used technique for cell segmentation in histological images is image thresholding ([28]). It follows the same principles explained above for noise reduction. The first and main hypothesis for this thresholding technique to accurately segment cells is that the colour intensities (or, more generally, properties) of the cells are radically different than the ones of the other image objects (as it's supposed to be the case when working with stained histological images). Then, the idea is to use this information to isolate image objects whose colour properties ensure that they are cells. As mentioned in [27] and as observed in our own experiments, the main assumption doesn't hold and thresholding on its own can't really lead to good cell segmentation results. In order to improve the effectiveness of such thresholding techniques, several pre- and post-processing tools can be used.

  As the pre-processing is concerned, we can use some previous morphology operations in order to take better benefit from the contrasting assumption. Noise reduction is one of those tools but it can be simpler processes as eroding, contrasting, smoothing or blurring operations.

One of the main drawbacks of the thresholding technique is that it's perfectly able to coarsely detect cell regions but it generally fails at detecting overlapping cells. One common solution to this problem is the watershed algorithms ([29]).
If anything, the watershed algorithms are usually good at segmenting cell agglomerates but their general drawback is still the over-segmentation, which means that some potential cell-candidate objects are cut in pieces. Over-segmenting is a side effect that we want to avoid. Once again, some of the previously mentioned pre-processing techniques can be used to reduce this over-segmentation effect.
Once the thresholding process has been executed, some too small or too big objects can remain in the resulting processed image. With the general dimensions of a specific cell type given, a size filter can be applied to get rid of noise or other artefacts.

- **Blob Detection:** as seen in [30], [31], [32] and [33], blob detection is a popular object detection technique which can be applied for the purpose of cell segmentation. First of all, let us define what blobs are and blob detectors. Lindeberg in [34] gives this definition of a blob: a [image] region associated with at least one local extremum, either a maximum or a minimum for respectively a bright or a dark blob.. More intuitively, a blob is an image area whose intensity characteristics are homogeneous and, in this work, we can associate the graphical occurrences of cells as blobs. A blob detection process, or blob detector, aims to detect such regions in an image.
  The way blob detectors work can vary from one implementation to another since there are various methods to produce them. Here's, taken from [31] the different approaches to blob detection:

  - *Template matching:* A fast and robust method for detecting blobs is template matching. A template of a blob is moved over the search image and blobs are detected where the template matches a part of the image. The main problems with this approach are, on one hand, the lack of flexibility if only a few templates are given as input of the blob detection process and, on the other hand, the high computational costs if we are to consider less templates but with the possibility of rotating them ([35]).

- *Watershed detection:* The watershed method assumes an image to be grey value mountains and simulates the process of rain falling onto the mountains, running down the mountain range and accumulating in basins. This process is repeated until all basins are filled and only the watersheds between different basins remain. These watersheds correspond to bright blobs, whereas dark blobs can also be obtained by taking the gradient amplitude image. The main drawback of this approach is the over segmented results (as mentioned previously in another context) so image pre-processing must be involved or raw, over-segmented, blobs must be merged ([36]).

- *Spoke Filter:* An early (single scale) blob detector which detects blobs of various sizes is the spoke filter, also called Adaptive Spatial Erosion Filter, is applied as follows:
  - Apply edge filters to extract local edge elements of all (8) orientations.
  - Mark pixels as "interior", which lie within a certain distance of an edge element on a line perpendicular to the edge tangent direction.
  - Mark spoke crossings as being interior pixels marked by edge elements of different orientations.
  - Mark blobs as being crossings marked by 6, 7 or all 8 directions.
  - By varying the distance, blobs of various sizes can be detected. ([37])

- *Automatic Scale Selection:* The principle for automatic scale selection is: in the absence of other evidence, assume that a scale level, at which some combination of normalized derivatives assumes a local maximum over scales, reflects the size of the corresponding blob. Scale levels are obtained by Gaussian smoothing. The Gaussian function meets the requirement that no details are generated when resolution decreases and it provides simpler pictures at coarse scales. One problem with this approach, and this is due to the Gaussian blurring applied before detecting any actual blob, is that blobs may be detected at coarse scales, and the localization properties may not be the best. Therefore a coarse-to-fine approach is needed to compute more accurate localization estimates. ([38])

- *Sub-pixel precise blob detection:* This technique is built upon Gaussian scale space. The blob is defined as a rectangle with constant contrast, which becomes a local extremum under sufficient Gaussian smoothing. Because this approach is applied to images containing equally aligned objects, like windows or cars, a rectangle is used as basic model of the blob. Also, some benefits arise from this technique: this method provides a way to construct the boundary of the blob and approximate the boundary by an ellipse. For blob classification, it can extract attributes like the blob's boundary length, area, geometric moments and the parameters of the fitted ellipse. ([39])

- *Effective maxima line detection:* this is a method where connected curves of modulus maxima at different scales - called maxima lines - are effectively selected, to divide blobs from noise. ([40])

- *Confidence Measurement:* this method won't be described in this section as it is very complex and not commonly used. It is not based on linear

Gaussian smoothing, as the previous methods, but on a complex algorithm. ([41])

One of the main reasons for the use of blob detectors while attempting to locate cells in an image is that it requires by far less user interaction than traditional (manual, semi-automatic or automatic) analysis methods. As shown in [33], it can even have a number of advantages over the latters if the cells are very similar to each other's (we are mainly talking about the size and the shape of the cells here): simplicity, applicability to a wide range of microscope images and optimized automation. The main problem with common blob detectors lies in the fact that they need some calibration step before use, they can require the average size or distance between nodes parameters. Also, they won't work effectively if the cells are not very similar (different colours, shapes or sizes, even if, for the last case, multi-scale blob detectors with automated scale selection can be implemented quite easily (see [30])). Finally, there are only but a few colour-based blob detectors, most of them are grey-scale blob detectors and thus can't benefit from colour properties which are very important.

- **Morphological filtering:** another popular approach to cell segmentation is directly connected to the field of mathematical morphology. Mathematical morphology is the field of mathematics concerned with the analysis and processing of geometrical objects. Some operators coming from this particular field, as erosion, dilatation, opening, and closing are well suited for the analysis of structural and topological properties of objects in images. They are then often used for the purpose of cell segmentation. Obviously, it is possible to build advanced filters by combining the applications of such operators into integrated filters. That is easily feasible and flexible to the needs of a particular image analysis. Once again, most of the morphological operators work on binary or grayscale images and either a pre-processing step or another type of approach has to be considered if one wants to take advantage of the colour properties of the image objects.

- **Region Accumulation:** the basic idea behind this approach is to consider some seeds in the image, which could be a random process or not, and then start to add the neighbouring (connected) points in order to form regions of similar characteristics, this is what's commonly called region growing. This approach suffers from the same drawbacks as the thresholding approach as it considers a pre-defined set of values of the analysed characteristics. Coupled with machine learning algorithms, this drawback can partially be avoided. Watershed segmentation, classified before as a blob detection technique, can also be considered as a region accumulation method.

- **Deformable Model Fitting:** this approach adopts a more analytical point of view. That is to say that objects will be only considered as satisfying or not satisfying some defined constraints like a parametric contour (2D) or surface (3D), or even, more implicitly, the zero-level of a specific function. ([27]) Initialized by a first, coarse segmentation, a deformable model is iteratively evolved to minimize a predefined energy functional, which typically consists of image-based and shape-based terms, the latter of which allow the incorporation of prior knowledge to constrain the results. The critical point in this approach is obviously the choice of the function used for object evaluation. That said, this approach performs well as topology changes are concerned because of the parametric aspect of the defined constraints.

ii. Contour-based approaches

- **Manual segmentation:** this approach is the easiest contour-based approach. The user has to select some boundary points of the cell and a curve is drawn based on these. Far to be an automated process, this requires an extensive user interaction and is designed to be used for a limited scope analysis. For a large number of cells, manual segmentation is not often considered but it can serve as a starting point (seeding) for a more automated technique ([42], [43]).

- **Active contour ("Snakes"):** this approach is also based on boundary points but their detection is then performed automatically. A snake is an energy-minimizing spline guided by external constraint forces and influenced by image forces that pull it toward features such as lines and edges. Snakes are active contour models: they lock onto nearby edges, localizing them accurately ([44]). In active contour models, the accuracy of a final boundary mainly depends on the initialization of the contour points. However, this task is not easy and often requires user interaction ([8]).

iii. Hybrid approaches

More recently, the previous approaches have been both used in some experiments, as it's the case in [45], where the authors designs an algorithm which combines cell boundary evaluation and region growing technique for cell cluster segmentation. Experimental results show an improvement of 33.33% in the reliability and an improvement of 55.1% in the accuracy of the cell cluster segmentation results.

iv. Discussion about cell segmentation techniques

In his overview of the cell segmentation techniques, Erik Meijering discusses about the evolution and the applicability of those ([27]):

Several observations follow from the analysis of the literature on cell segmentation in the past 50 years. First, most of the different approaches were originally developed for applications in other fields (computer vision, robotics, materials science, medical imaging), and were later adopted for cell segmentation. This is remarkable, given the unique and unparalleled challenges in cell image analysis, which should provoke the development of original ideas. Second, even though new approaches are introduced once in a while, they seem to never fully replace old ones. Apparently, while none of them alone produces satisfactory results, they all continue to be useful to some extent. Third, as a consequence, methods proposed in recent times are rarely based on a single new concept, but are often merely new combinations of the discussed approaches, tailored to a specific application. Rather than converging to a robust, unified solution, it thus seems that the field is diverging, and by now almost as many cell segmentation methods have been developed as there exist cell analysis problems. However, the explosion of technical papers in the past decade suggests that the performance and applicability of these methods remain limited, and more powerful methods will need to be developed.

He also talks about the future of cell segmentation and his analysis is both alarming and full of hope for the development of this particular field. If we take a look at what has been performed in the field in the last 50 years, what we can mainly observe is that lots of techniques have been developed but none of them has been able to replace another or become the reference technique. Instead of being generic and fitted for a wide range of histopathological images, they are generally suited for a very specific set of images and can't really be reused. Another problem is also the way the techniques can be great and really overcome some critical problems in cell segmentation but they are not very well documented, only introduced in scientific papers. A way of overcoming this problem would be to integrate those into an open-source framework for image analysis or cell detection (ImageJ, Cell Profiler, etc.). It's also quite hard to determine if this problem is just a technical one or if, even with the most advanced materials, cell segmentation will remain a notoriously difficult problem. Finally, the need of automation in medical processes should eventually lead to a boost in innovation for cell segmentation techniques. Nowadays, most of them are not original ideas but the result of the adaptation of image analysis techniques coming from other fields or domains.

B. Custom processing pipelines

In this section, we will introduce four of our different attempts to come with satisfactory image processing pipelines for cell segmentation. There were many of those attempts as we discovered the design of such pipeline in the very beginning of this project but the 4 pipelines that we will describe below are the ones that lead to the best results and adopt enough different approaches to be introduced in this work. The three first pipelines are the result of our first attempts at designing an image processing pipeline. Thanks to that, the design phase has finally matured in something we can consider as quite satisfactory. Moreover those pipelines don't require any user interaction but they considerably differ from one another either conceptually or practically. On the other hand, the fourth pipeline came rather late in our design process, suffers from a lack of maturity and robustness, and relies on (non-intensive) user interaction. However, it relies on a common but yet innovative technique that we would like to introduce in this work and it would certainly get better if it was refined in future work.

a) Image processing pipeline 1: morphological image filtering

i. Pipeline overview

Figure 5 displays a high level view of the processes involved in the first pipeline for cell segmentation.

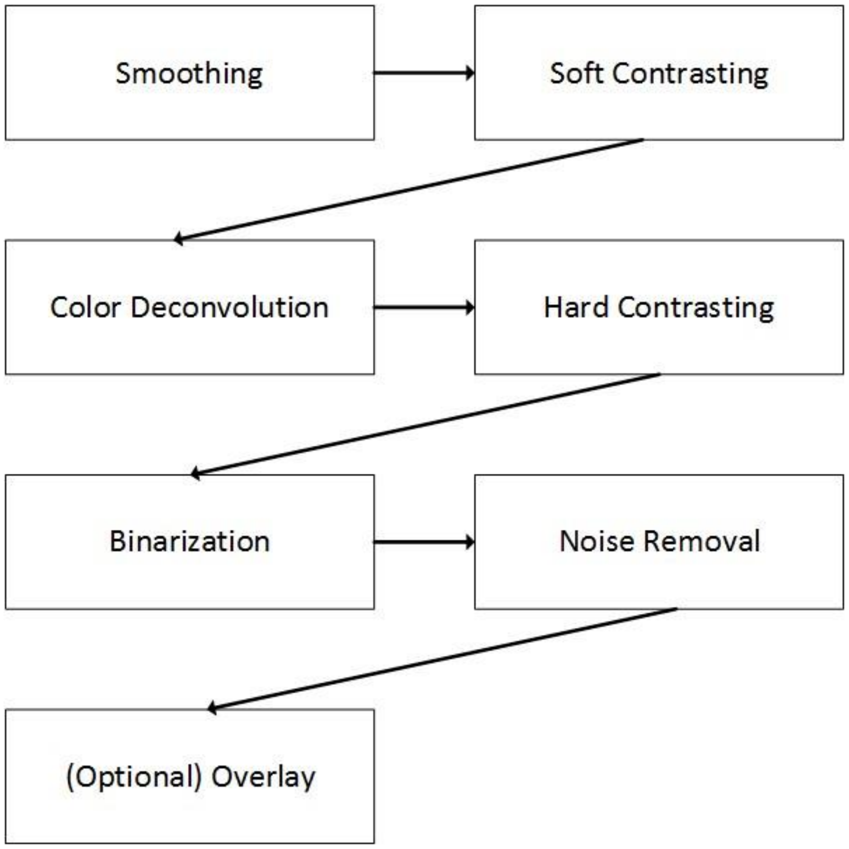


Figure 5 - Overview of our first processing pipeline for cell segmentation

We will describe each of these processes following the sequence in the pipeline and also explain why we chose those particular implementations of the processes over others.

- **Smoothing:** as previously mentioned, image smoothing is among the most common techniques used for image pre-filtering. Smoothing is an image processing technique that aims to attenuate noise in an image and can even lead, to some extent, to detect edges in an image ([46]). There are several approaches to perform smoothing on an image and we tackle the problem with two major techniques: anisotropic diffusion and Gaussian filter. The reason why we start the pipeline with this smoothing process is that raw stained images can contain noise that we want to eliminate as following techniques will be sensitive to noise. Noise can be roughly defined as anything we are not interested in the image and can be caused by multiple factors: light fluctuation, sensor noise, quantization effects, finite precision, and so on ([47]). The two techniques have been implemented within the ImageJ framework. The Gaussian smoothing filter is implemented in the core packages of the platform and we had to customize our own anisotropic diffusion filter.

  - *Gaussian smoothing filter:* the Gaussian smoothing technique allows to reduce noise without losing too much detail (remove important image gradients). First, the central pixels (points where there are major image gradients) are identified and then, a weighted averaging process is engaged in order to assign a new colour intensity to the neighbours of central pixels. This weighted averaging process allows to focus on central points (edges) and to give less weight to the neighbours ([47]). The Gaussian smoothing is available in ImageJ by just clicking on *Process > Smooth*. For the purpose of our custom application, we have used the ImageJ Java archive and import the functionality into our application. Figure 6 shows the effect of the application of such filter.

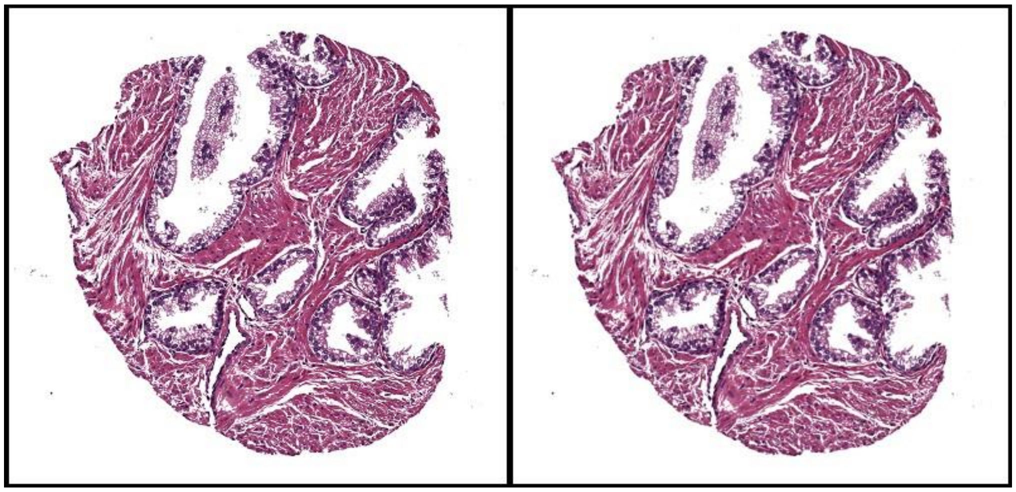


Figure 6 - Gaussian smoothing filter applied twice on an H&E stained prostate image

- *Anisotropic diffusion filter:* the model of anisotropic diffusion on top of which we rely for smoothing purpose was introduced by Perona and Malik in 1990 ([48]). We won't define mathematically the diffusion process or neither the anisotropic diffusion process itself as it involves complex concepts that we think are not relevant for the purpose of our work. Still, the basic idea of this technique is to generate a parameterised family of more and more blurred images from the initial one with a diffusion operator (that's the technique used for blurring). Each image in the set of blurred images is convoluted (mixed with a convolution operator) with the initial image and then this process is repeated several times until completion of the algorithm. In our application, we reused an ImageJ implementation of Vladimir Pilny and Jiri Janacek but we customized it to our needs in the application. The main drawback of this technique, despite its accuracy, is that it is really slower than the Gaussian filter technique. That's the reason why we finally went for the first approach in our pipeline (within the same environment setup, the Gaussian filter technique takes about a second to be run, the anisotropic diffusion process takes about 20 seconds without enough accuracy increase to justify its use in our application). Figure 7 shows the effect of the application of such filter.

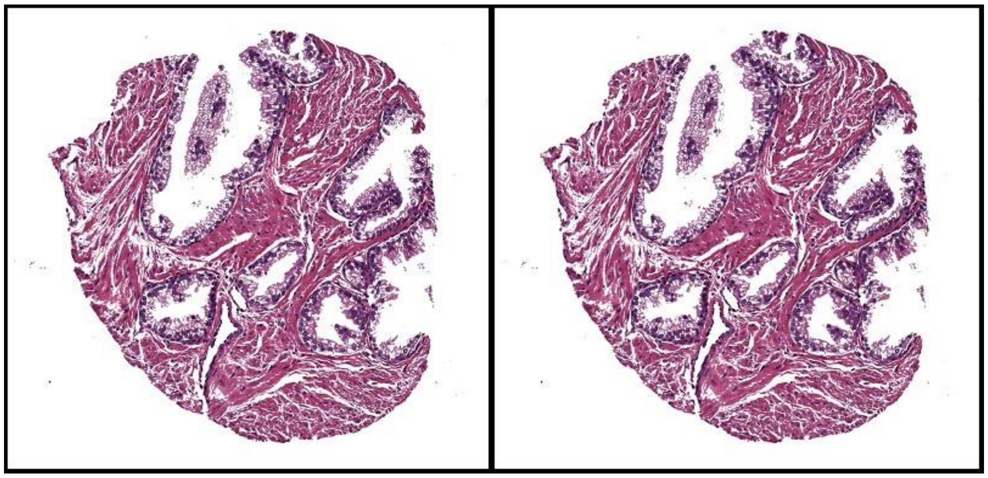


Figure 7 - Anisotropic Diffusion Filter applied on an H&E stained prostate image

- *Soft Contrasting:* one other common pre-filtering technique is contrasting. The idea behind the use of this filter was to enhance the contrast between the blue and red elements in the H&E stained images and to take advantage of this enhanced contrast for the next stage of the pipeline, which is colour deconvolution. Once again, there are several methods to perform image contrasting but our goal at this stage was to stay close to the true colours of the H&E stained image while making it easier for the next process to perform. That's why we talk about soft contrasting. One of the first contrasting method we used was a technique based on human perception of contrast because we figured that's what allowed doctors to distinguish between cells and other image artefacts. This method has been developed by Arini and Majumder ([49]) and we adapted their implementation to our Java application. The method relies on the fact that human contrast sensitivity increases proportionally with the increase in the mean local luminance and makes use of this knowledge to enhance contrast in an image by controlling its gradient with a single parameter ([49]). For this particular technique, we didn't rely on ImageJ, the technique was embedded in a standalone java module in our application.

Note that this method requires a parameter Δ to be set, this parameter roughly lowers the intensity of the perceived contrast and, after multiple tests and attempts, we determined that $\Delta = 3$ was a good compromise between performance and the goal we wanted to achieve with the method.

Figure 8 shows the effect of such contrast enhancement method on an H&E stained prostate image.

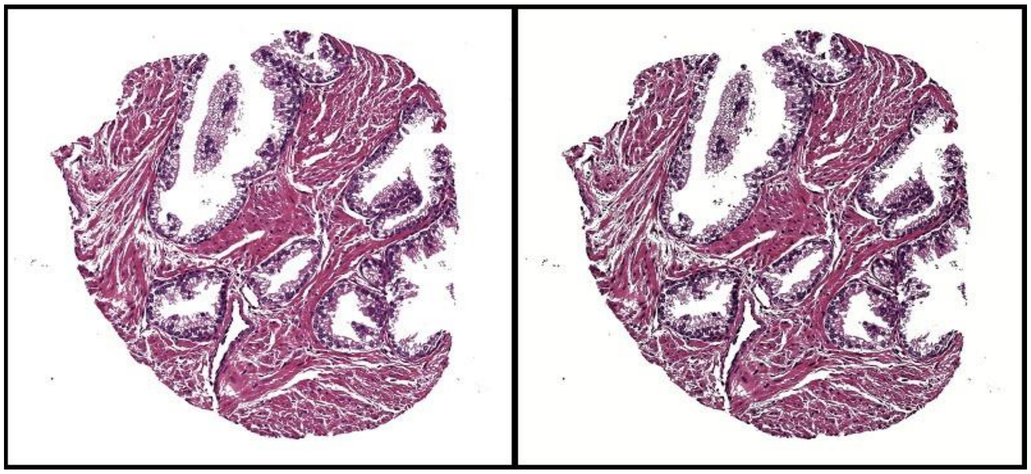


Figure 8 - Perception-based contrast enhancement filter applied on an H&E stained prostate image ($\Delta = 3$).

The main drawback of this method is its running time. Indeed, and even with local optimizations performed by the authors in the Java source code, it took between 29 and 34 seconds on a standard laptop machine (Dual core processor 2.5GHz (4 logic cores) and 4Gb of RAM) for each image (1603 x 1603 pixels) to be analysed whereas other techniques with comparable performance achieve far better execution time results. That's the main reason why we didn't stick with this method (as it was similarly the case with anisotropic diffusion for the smoothing stage).

We then decided to exploit the embedded contrasting process in the ImageJ framework which is based on an image histogram stretching. An image histogram is a graph representing the colour distribution of the pixels in the image. The x axis of such histogram represents the colour value (e.g. for a classic RGB image, we would have red, green and blue histograms, each with 256 tonal values ($[0 - 255]$) for the x axis) and the y axis represents the amount of pixels in for each colour value (colour value frequency). Histogram stretching is one of the most basic image enhancement techniques and attempts to enhance the contrast of one image by literally stretching the histogram of the image. Stretching the histogram means that, based on the determined range of colour values, the algorithm will stretch the initial colour values to span the defined colour range. In order to define this range, we have to set its lower and higher bounds. Most commonly are taken the minimum and maximum colour values of the image as those bounds. The ImageJ contrast enhancement process also allows to set a saturated pixels parameter which represents the percentage of pixels that we allow to become saturated after the execution of the process (a saturated pixel has a very high tonal intensity value) ([50]). In our experiments, we determined that allowing 0.5% of the pixels to become saturated gave the best results for the purpose we wanted to achieve at this stage of the processing pipeline. Once again, this technique makes us benefits from its short running time which was around 900 milliseconds on our test machine (1603 x 1603 pixels images).

Figure 9 shows the effect of such contrast enhancement method on an H&E stained prostate image.

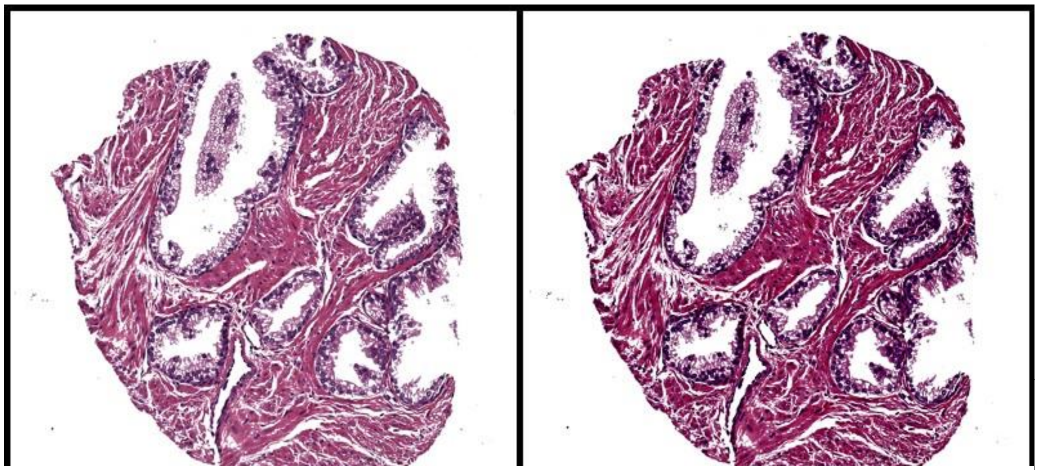


Figure 9 - Histogram stretching based contrast enhancement filter applied on an H&E stained prostate image.

- *Colour Deconvolution:* the third process in this processing pipeline is one of the most commonly used tools when designing cell segmentation engines working with colour images. Indeed colour deconvolution is a colour channel segmentation technique that allows to highlight image objects in each colour channel based on their colour intensities. More precisely the goal of this technique is to separate the relative contribution of the dyes (in our case, the images are stained with H&E dyes) to the resulting absorption spectrum, or the perceived colours ([51]). The design of a colour deconvolution tool is certainly not trivial but fortunately, as it isn't a new field of research, several implementations of this method can be found in the literature.

  As for the chosen implementation of the colour deconvolution method in our project, we re-used an ImageJ plug-in designed by Ruifrok and Landini in order to integrate it into our Java application. This implementation is the direct transposition of the technique described in [51] and works for a lot of staining schemes. Of course, we set the parameters in order to fit our set of H&E stained images. This implementation of the colour deconvolution technique runs within a reasonable amount of time with an average execution time around 2.5 seconds (1603 x 1603 pixels images).

  Figure 10 shows the effect of such colour deconvolution method on an H&E stained prostate image.

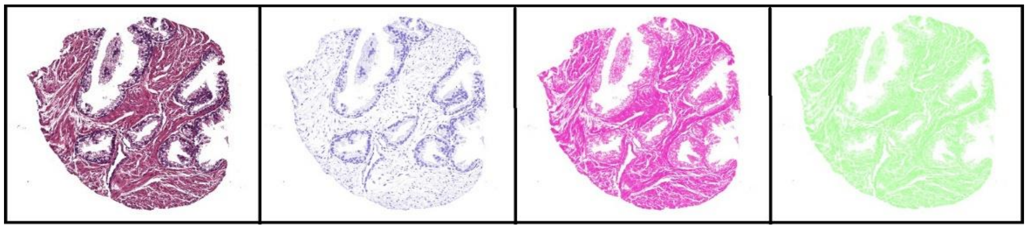


Figure 10 - Colour deconvolution filter applied on an H&E stained prostate image.

As we were designing a cell segmentation pipeline, we were particularly interested in the blue channel because cell objects are coloured in blue thanks to the H&E staining protocol. That's why we focused the following processes onto the blue channel image resulting from the colour deconvolution process.

- *Hard Contrasting:* as you can see in Figure 10, most of the cells in the image are detected at this stage of processing in the pipeline. However if we, as humans, can already distinguish the cell structures and the cells themselves, it's still hard for our next step of computing to work on this kind of images and to extract useful features. The idea at this stage of processing is then to be able to really highlight the cell objects and then store some information about them for further processing. With this idea in mind, we applied what we call a hard contrasting filter to the blue image coming from the deconvolution process. The goal was really to get rid of the noise surrounding the cell structures. The kind of contrasting filter we used at this point is based on a look-up table modification for the image. In image processing, a common look-up table (LUT) typically called the palette allow to determine the colours used to display the image. With specific manipulations of this table, it is possible to change the brightness or the contrast of an image or even to get its negative image. We used that technique in our application for this stage of processing.

Figure 11 shows the effect of such hard contrasting method on an H&E stained prostate image.

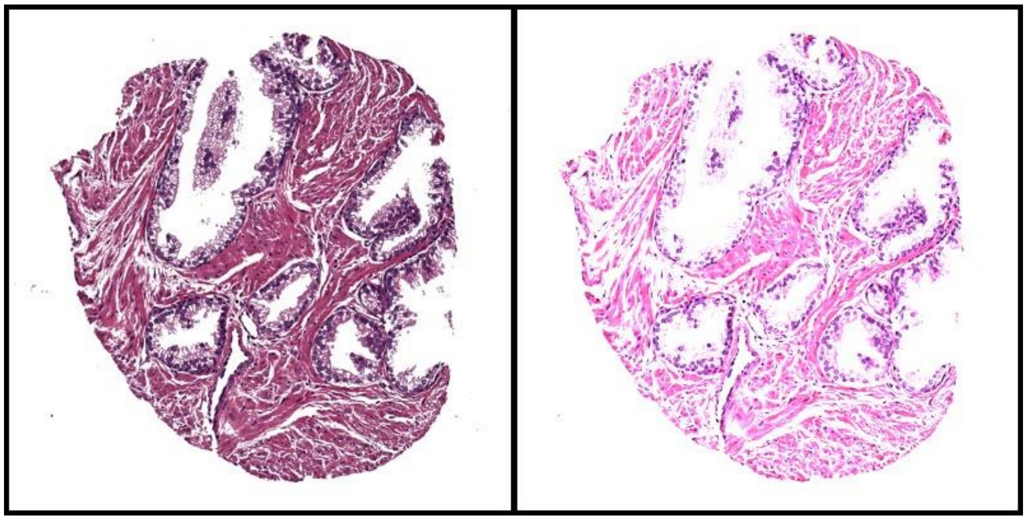


Figure 11 - Hard contrasting filter applied on an H&E stained prostate image.

- *Binarization:* once we apply the previous filter in our processing pipeline, most of the cell objects have been identified but some light noise remains in the image. In order to remove this kind of noise, we apply a basic binary filter which will transform the image into a black and white one (no grayscale here, pure binary image). With this filter being applied, we can be sure that only what we consider as cell objects are the only (expected) kind of objects in the resulting image (those objects would then be the black ones). We developed a first algorithm which would detect the edges in the given image and then fill the holes of the detected contours but we finally used a simpler method, which is the embedded Make Binary process in the ImageJ framework. We don't show in this paragraph the result of applying such filter as we consider this as rather trivial.

- *Watershed:* as previously mentioned, the watershed technique can help with the natural segmentation of objects in a given image. One of the problems we faced at this stage of the image processing was that aggregates of cells couldn't have been segmented the way they should have in order to detect individual cells instead of aggregates. To overcome this problem we decided to try and see what could be the benefits of such technique at this stage in our processing pipeline. As the ImageJ framework offers an implementation of such method, we decided to use this implementation in our Java application.
  We didn't really experience any over segmentation effect on the majority of the images we had in our image set and the implementation has the advantage of running quite fast with an execution time below one second (1603 x 1603 pixels images).
  Figure 11 shows the effect of such watershed technique on an already binarized H&E stained prostate image.

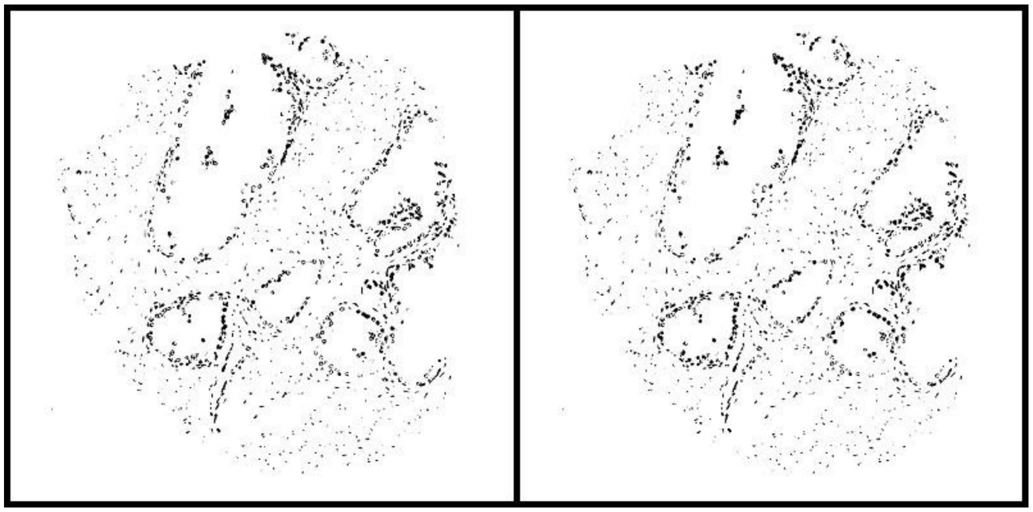


Figure 12 - Watershed technique applied on an already binarized (stage 5 in our processing pipeline) H&E stained prostate image.

- *Noise Removal:* this step in the image processing pipeline allows us to remove even more noise and the algorithm for this noise removal process is rather simple. We first detect all the objects in the image (which is now simple because objects are in black and the background is white) and then we remove the smallest objects with a defined threshold for size which will be the lower bound for image object size. We implemented this noise removal process on basis of the flood fill algorithm in our application. This algorithm originally serves the purpose of detecting connected areas in an image. We won't describe here all the details of our implementation (consult [52] for more) but we used a 8-way version of the algorithm, which means that each time we go over a point in the image, we are looking in 8 directions (north, north east, east, south-east, south, south-west, west and north-west), and we also made used of a loop storing the to-be-explored points in LIFO to avoid too many recursive calls or memory overflow issues during the execution of the process. After several attempts on a set of about 20 different prostate images, we defined the size threshold with a value of 10 pixels, which means that the size (the number of pixels) of an image object couldn't be lower than 10 pixels (this value depends on the size of the images in the test set and is parametrisable). Otherwise, the object was considered as noise and removed from the image (set to the white colour value). We are well aware that setting the size threshold manually makes this process dependent on the input images and especially their size or resolution. Further improvement of this process should dynamically set this threshold based on the object records find in the image.

- *Merging:* this last mandatory process only uses the processed image as an image filter for the initial image. That's how we can keep the true colours of the initial image in order to perform the colour analysis in the second computing step of our diagnostic engine.

Figure 13 shows the effect of such merging filter on an H&E stained prostate image.

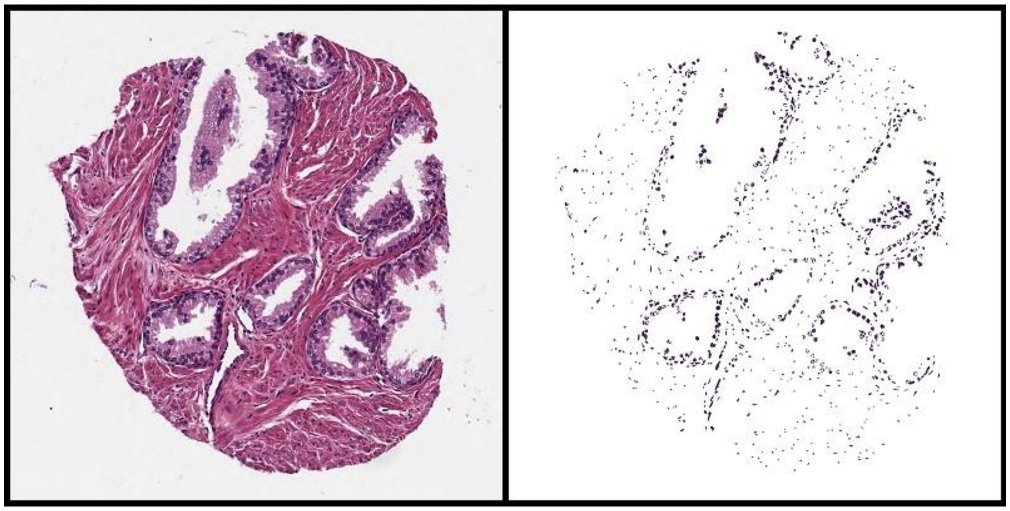


Figure 13 - Overlay filter applied to H&E stained prostate image after cell segmentation with the first image processing pipeline.

- *(Optional) Overlay:* the last optional process will project with a specific colour the result of the cell segmentation by our image processing pipeline on the initial image.

  Figure 14 shows the effect of such overlay filter on an H&E stained prostate image.

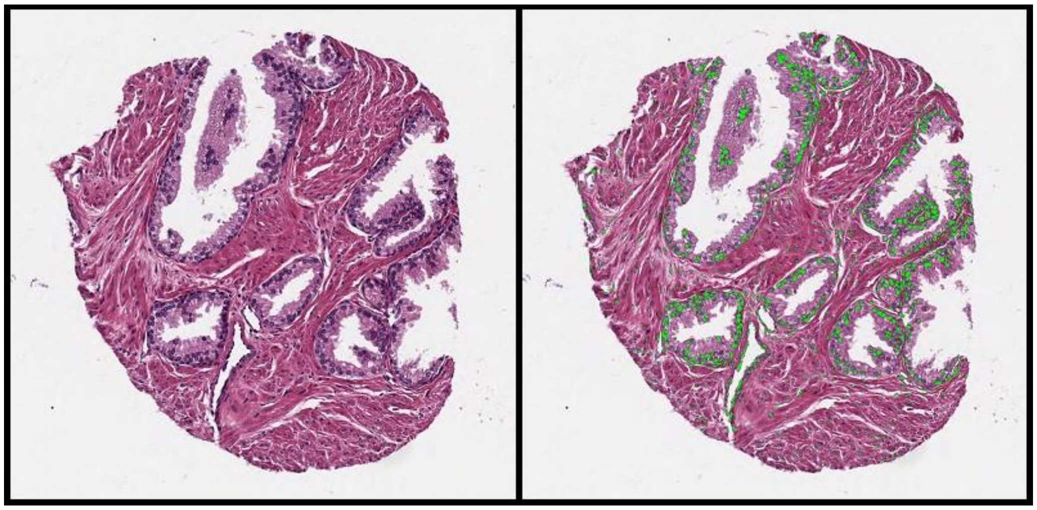


Figure 14 - Overlay filter applied to H&E stained prostate image after cell segmentation with the first image processing pipeline.

Figure 15 shows the successive stages of our first processing pipeline with an H&E prostate image as input.

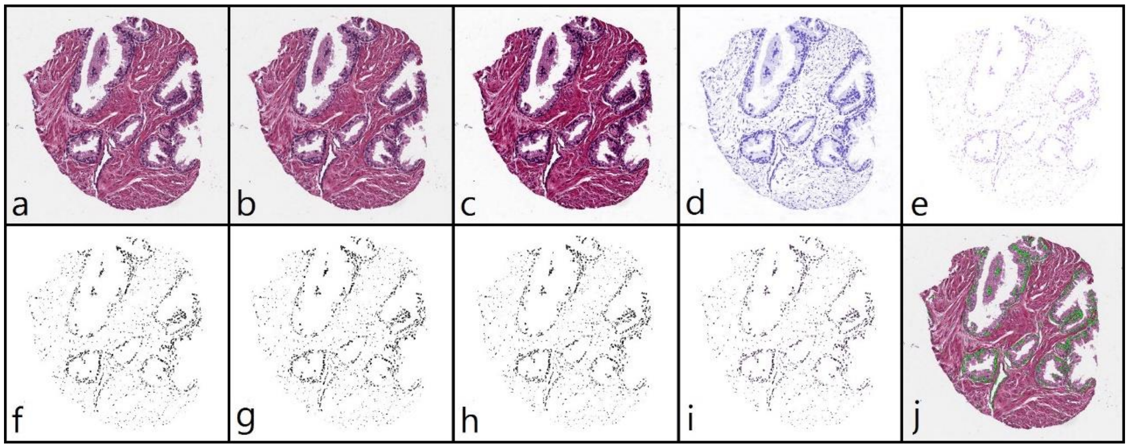


Figure 15 - Each image is a result of a processing stage in the first image processing pipeline. a) Initial image, b) Smoothing, c) Soft contrasting, d) Colour deconvoluting, e) Hard contrasting, f) Binarization, g) Watershed, h) Noise removal, i) Projection, j) Overlay

b) Pre-processing pipeline 2: H&E clustering based pipeline

i. *Pipeline Overview*

Figure 16 displays a high level view of the processes involved in the second pipeline for cell segmentation.

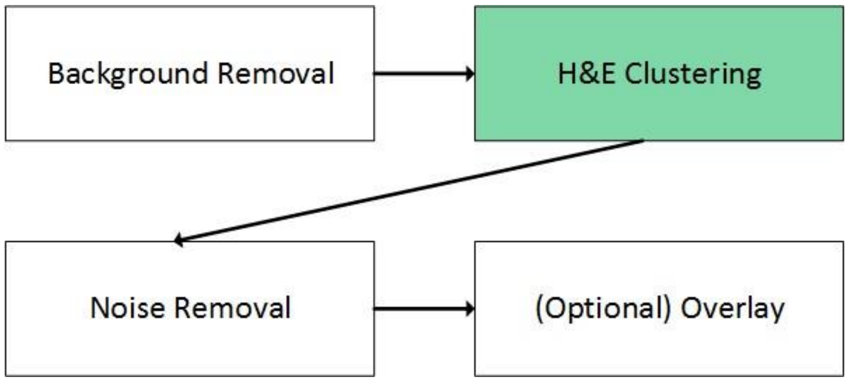


Figure 16 - Overview of our second processing pipeline for cell segmentation

This second pipeline reuses some of the processes of the first one but is conceptually less complex to introduce. Indeed the basic idea of this pipeline is to use a clustering technique that will make groups of pixels in the image based on their colour. Once again, we take advantage of the colour properties of the image set we work with and we produce a two-phase pipeline: image preparation and H&E clustering. The image preparation consists of background removal in order to optimize the running time of the second process which will then ignore white background pixels. This background removal is based on a RGB clustering method which ends by defining two groups of pixels: the background ones and the foreground ones. In order to perform such colour clustering, we rely on a custom k-means clustering algorithm with specific seeding (black and white seeds). As H&E clustering is concerned, we also use this kind of custom k-means algorithm with a specific seeding. The choice of the seeds prior to the execution of the algorithm instead of randomly choosing the seeds for colour clustering is intended to give some direction to the groups that will emerge from the clustering process. We basically want 3 groups of pixels as results of the classification: background pixels, haematoxylin-stained pixels, and eosin-stained pixels. Thanks to appropriate colour samples in our image set, we choose the seeds as being white, blue and red with calibrated RGB values. After those two main processes, we use the noise removal process from the first processing pipeline, we use the processed image as a filter for the initial one and we can also optionally apply the overlay filter introduced for the first pipeline.

Figure 17 shows the successive stages of the second processing pipeline with an H&E stained prostate image as input.

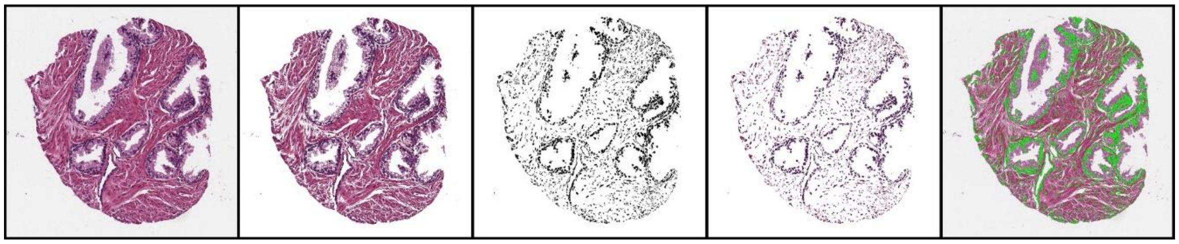


Figure 17 - Each image is a result of a processing stage in the second image processing pipeline. a) Initial image, b) Background Removal, c) H&E clustering d) Projection e) Overlay

If this image processing pipeline has for advantage to be conceptually less complex, its main drawback is its running time. Indeed clustering algorithm complexity depends on the size of the input set of values to cluster and with this kind of images, even if we performed the background removal in order to ignore white pixels, we still have large input to provide to the algorithm. Compared to the first pipeline that takes about 8 seconds to perform on a standard image (1603x1603 pixels), this algorithm takes 11 seconds on average for coarser results.

c) Pre-processing pipeline 3: rule-based colour segmentation

i. *Pipeline Overview*

Figure 18 displays a high level view of the processes involved in the third pipeline for cell segmentation.

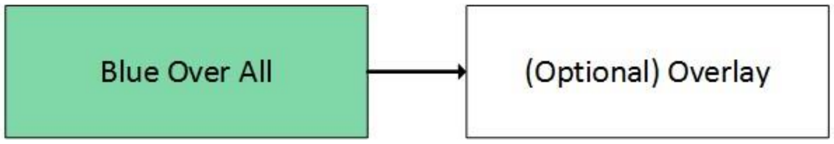


Figure 18 - Overview of our third processing pipeline for cell segmentation

This third image processing pipeline filters the image with only one rule based once again on the colour properties of the image. One simple information we can learn from the H&E staining phase is that the intention behind this staining is to distinguish image objects based on a specific colour attribution and we know that haematoxylin reacts with cell objects to produce blue shades in the image and that eosin reacts with non-cell objects to produce red shades. From this information, instead of running complex algorithm, we just considered that the pixels inside the cell-objects should be bluer than their non-cells counterparts. With this information only, we design a simple rule to apply to each pixel in the image: if the pixel is more blue than red or green and it isn't a background pixel (close to RGB(255,255,255)), let's consider it's a pixel belonging to a cell object. We first feared that this only rule wouldn't be restrictive enough to give good results but we were then really surprised after the first runs on a couple of images from our set. One of the notable feature we added to this filter is the ability for the user to set the difference between the red, blue and green colours prior to the execution of the algorithm as well as to define how close pixel colour intensities should be to a white pixel ones. A bigger colour intensity distance between colour channels will decrease the number of remaining pixels in the image and keep the objects that are really more red, green or blue than the others.

Figure 19 shows the effect of such blue over all filter on an H&E stained prostate image.

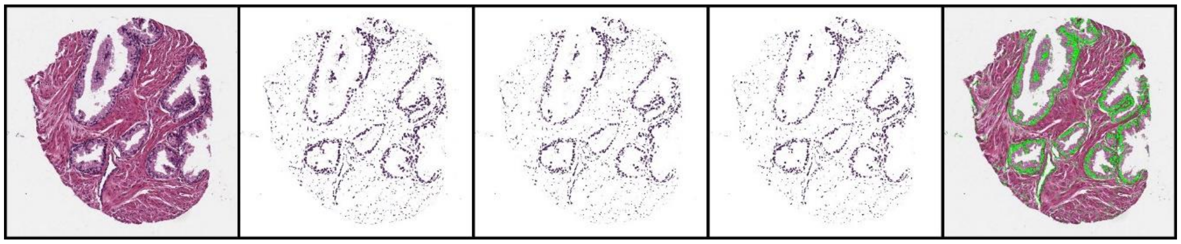


Figure 19 - Blue over all filter applied to H&E stained prostate image. a) Initial image, b) red difference = 0 and green difference = 0, c) red difference = -5 and green difference = 0, d) red difference = -5 and green difference = -5 e) overlay of (b) on (a)

The main advantage of this approach to cell segmentation is that it runs really faster than the previous ones (for the same settings, it runs approximately 8 to 15 times faster). Moreover the results we got were coarser but similar to what we had got previously with the other pipelines. Despite these advantages one could think that this approach is still the less generic one since it's tuned to our set of images. But it is not. Actually we have tested this algorithm on some other images and it proved to work pretty well if we chose the parameters of this algorithm accordingly.

Figure 20 shows the effect of such blue over all filter on an H&E stained breast image (out of our own image set).

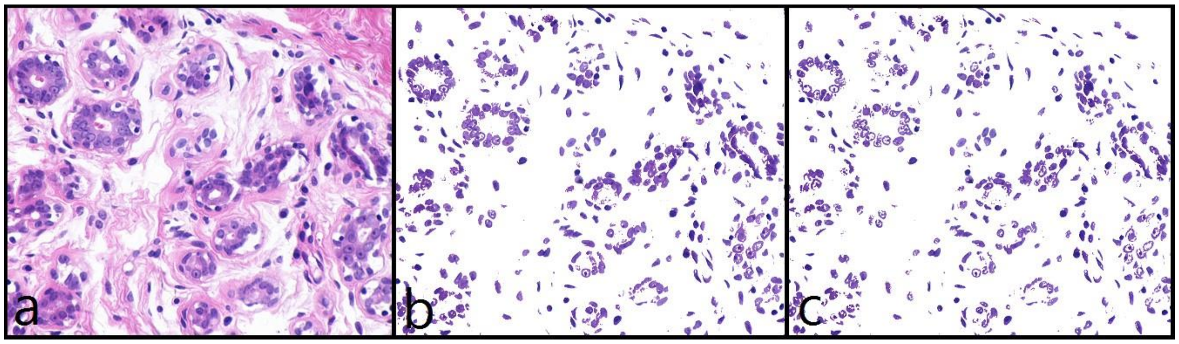


Figure 20 - Blue over all filter applied to H&E stained breast image. a) Initial image, b) red difference = -30 and green difference = 0, c) red difference = -40 and green difference = 0

Figure 21 shows the effect of such blue over all filter on an H&E stained breast image (out of our own image set).

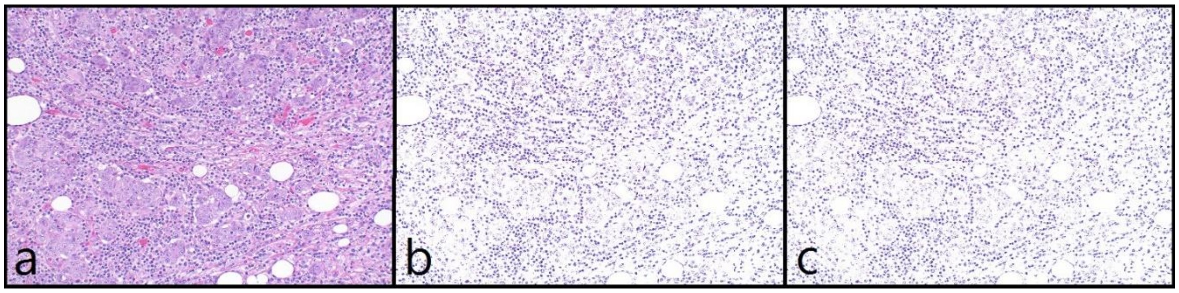


Figure 21 - Blue over all filter applied to H&E stained breast image (2). a) Initial image, b) red difference = -30 and green difference = 0, c) red difference = -40 and green difference = 0

d) Pre-processing pipeline 4: semi-automated cell segmentation

i. Pipeline Overview

Figure 22 displays a high level view of the processes involved in the fourth pipeline for cell segmentation.

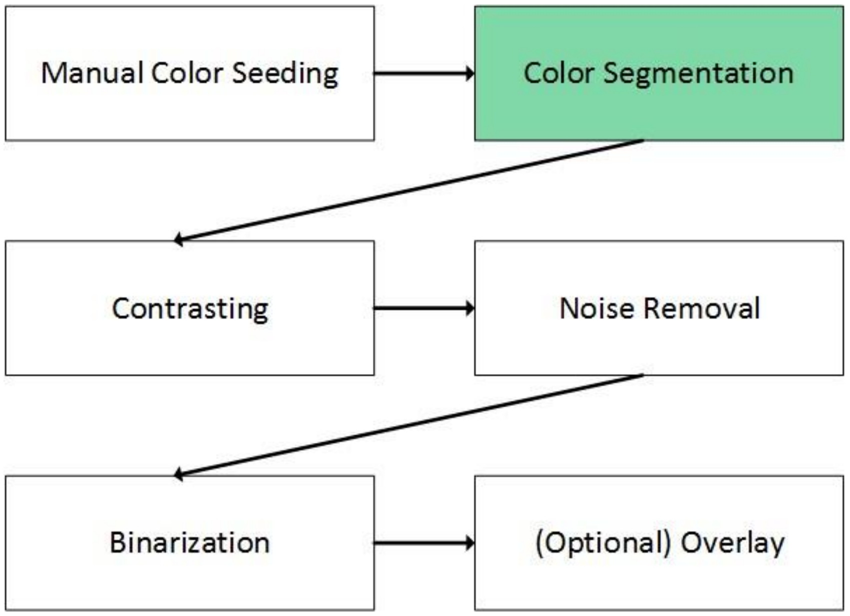


Figure 22 - Overview of our fourth processing pipeline for cell segmentation

The fourth and last image processing pipeline that we want to introduce in this work still relies on the colour properties of the images but involves more (that is, some) user interaction for the first process of its computing cycle. The idea with this pipeline was to be in an oncologist's shoes by thinking about how diagnosis was performed most commonly, that is rather subjectively with a human observation of cells and tissues to eventually come with a result in the form of a grade result on a specific scale (see Gleason grading section). Thus, we wanted to come the closest possible to human vision and how human interprets colour intensities, how a human perceives objects in an image, how an oncologist can ultimately distinguish blue and red objects and say with subjective but assured certainty that one object is close to another one from the colour point of view, that one image object is a cell and another one an artefact or a piece of tissue.

There is one colour space that allows to come very close to human perception of colours and also allows to introduce the concept of human-based colour distance. This specific colour space is the CIELab colour space. Before giving some basic explanation about this colour model, taken from [53], let us say that this colour space has been introduced by the CIE (International Commission on Illumination) especially to deal with the human perception of colour issue introduced in the introduction of this section.

"Colour space defined by the CIE, based on one channel for Luminance (lightness) (L) and two colour channels (a) and (b). One problem with the XYZ colour system, is that colorimetric distances between the individual colours do not correspond to perceived colour differences. The CIE solved this problem in 1976 with the development of the three-dimensional Lab colour space (or CIELAB colour space). In this model, the colour differences which you perceive correspond to distances when measured colorimetrically. The a axis extends from green (-a) to red (+a) and the b axis from blue (-b) to yellow (+b). The brightness (L) increases from the bottom to the top of the three-dimensional model."

Image 23 represents the CIELab colour space.

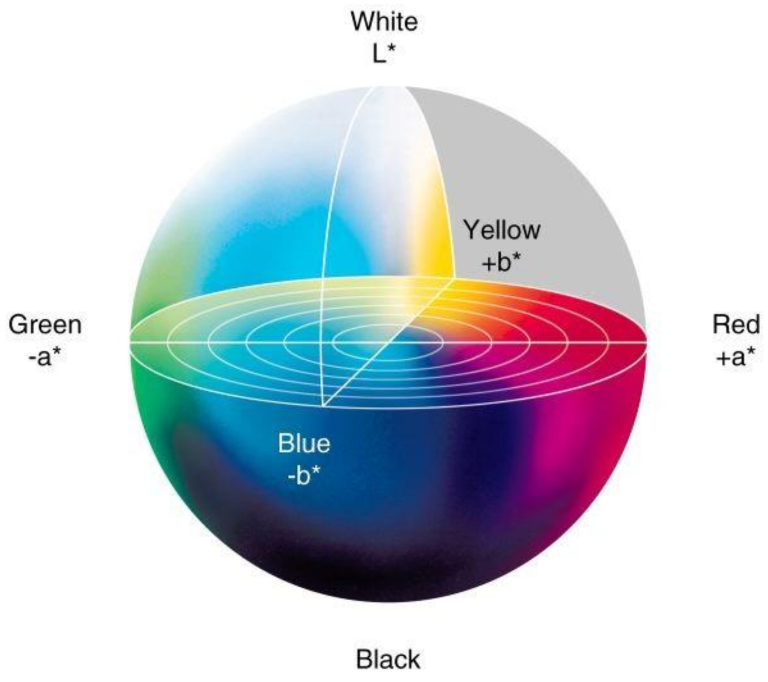


Figure 23 - CIELab colour space ([54])

"This colour space is better suited to many digital image manipulations than the RGB space, which is typically used in image editing programs. For example, the Lab space is useful for sharpening images and the removing artefacts in JPEG images or in images from digital cameras and scanners."

Having introduced basic principles of this CIELab colour space, we can now explain what we meant by human-based colour distance. We have to mention that there is no clear rule defining how a distance (e.g. Euclidean distance) in the RGB colour space can be correlated with the how a human will distinguish the colours separated from this distance (and that statement is also true for CMYK, HSV or HSL colour spaces). That's why we need to work with another kind of colour distance. One of the possible distances for this kind of purpose is the Delta E colour distance. Because humans don't perceive all the colours with the same sensitivity, the mathematical formula for the computation of this colour distance is an attempt at taking into account this kind of sensitivity difference. More information on this topic can be found in [55] but we won't go through more details in this work since it's clearly not our purpose.

From these two concepts, we have attempted to reproduce the human process of prognosis or diagnosis with the implementation of our fourth pipeline: first, we need that the user performs some colour sample in the image to get the seeds of our colour segmentation process which follows this seeding process. This last process will then take those seeds, generate the CIELab average value from this colour set and finally use the delta E colour distance from each pixel in the image to this average colour. If the distance is beyond a statically computed threshold (this is possible thanks to a universal agreement about delta E value scale and how it is related to human perception of colour), we can discard the pixel as belonging to an object of interest for the user. The other processes following this major one are noise removal, binarization and (optionally) overlay. Those processes have already been introduced for the previous pipelines.

Figure 24 shows the effect of such human-based colour segmentation filter on an H&E stained prostate image.

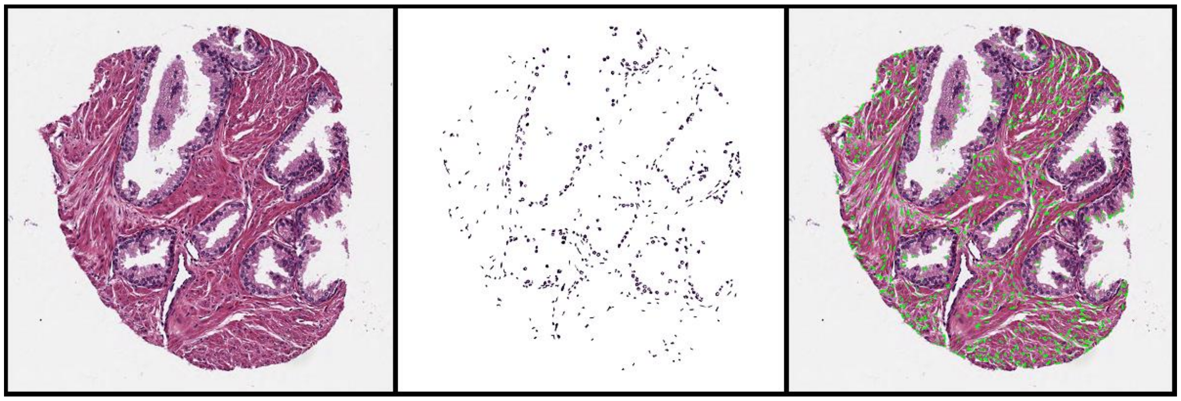


Figure 24 - Human-based colour segmentation filter applied to H&E stained prostate image: a) initial image, b) filtered image c) overlay

e) Discussion

In this section we want to compare the four introduced pipelines with regards to 4 criteria. We have selected 20 representative H&E stained images, 10 normal, 10 malignant.

i. *Running time*

- **Machine configuration for tests:** Asus U36SD (Intel i5-2410M, 4 GB RAM).
- **Image settings:** RGB images – 1603 x 1603 pixels.
- **Average running time:**
    - ➢ Pipeline 1: 2.016s (2.560s with overlay)
    - ➢ Pipeline 2: 6.988s (7.446s with overlay)
    - ➢ Pipeline 3: 0.561s (0.994s with overlay)
    - ➢ Pipeline 4: depending on the time that the user takes for the seed selection (generally around one minute).
- **Conclusion:** the third pipeline is by far running faster than the other ones. This is due to its trivial but still efficient design. The first pipeline, despite its conceptual complexity, still runs in acceptable time period. Pipeline 2 suffers from the complexity of the clustering algorithms involved in its design. Local optimizations or even the choice for another, more performant, clustering algorithm should decrease this running time. Finally, pipeline 4 relies on some user interaction and is consequently the slowest technique for cell segmentation.

ii. Accuracy

- **Meaning:** practically, we would have needed the help of an oncologist team to assess about the accuracy of our pipeline but it hasn't been the case since it should have required more time and more resources that our project would allow. Moreover, we couldn't have a list of all cells in each image, even if it would have been helpful to measure precisely the accuracy of our pipelines. However, thanks to the precious advice from our supervisors, we gained sufficient knowledge of cell architecture and constitution to be able to distinguish *humanly* the cell objects in an image.
- **Results:** from our perspective, our pipelines consider only but a very few false positives for cell detection. On another hand, there are some differences about how the cells are detected with our pipelines. If the first pipeline is rather *shy* at detecting cell objects (by *shy* we mean that the amount of true positives can be sometimes low, especially when the initial H&E staining hasn't produced a satisfactory contrast between cell objects and non-cell objects), the second runs with completely opposite results and gives coarser objects, with several aggregates instead of individual cells. The third pipeline, once again, despite its simplicity in design, proves to give very satisfying results with more true positives than the first pipeline and less coarse results than the second pipeline. Finally, the fourth pipeline can give very satisfactory results when an appropriate seeding is performed but proves to produce poor results when the seeds are not well chosen.

iii. Genericity

As it was one of our purposes at the beginning of this work, our four pipelines benefits from a high genericity factor and proved to work on a rather wide range of H&E stained images (not only prostate images but also breast images e.g.).

iv. Conceptual complexity

Once again, the third pipeline is probably the easiest pipeline to design and to understand since it involves a single rule for all the image filtering. On the contrary, the first pipeline could be seen as quite complex since it involves multiple processes but each process doesn't imply so much complexity. As for the second and fourth pipelines, the former is based on only one concept, which is colour clustering, and the latter is probably the closest to what humans usually do in this kind of situations and is then quite easy to understand as well.

v. Future fixes and improvement

One major improvement that could be performed for all pipelines for better accuracy is the use of a template matching process which would take the true positives detected in a first pass of the pipeline and then look for similar objects that would have been left over during this first pass.

Local optimizations for performance could be also performed for all pipelines as we don't have much experience in the tuning of such algorithms. However, as mentioned earlier, most of our implementations are taken from well-known implementation or just re-used from an image processing framework (*ImageJ*). Finally, as coarse results are concerned, segmentation processes could be involved in order to avoid large agglomerates of cell objects.

4. Feature extraction and selection

A. State of the art

Once the input images have been processed in order to extract the potential cell objects, the logical next step in each automated diagnosis engine is to analyse those objects and to select the features whose values will allow to characterize different groups of cells and further to support the automated diagnosis process. It's also common to first define the type of features we want to work with and then choose a feature selection policy.

a) Feature type

As introduced in the image processing section, the automated diagnosis engine aims to detect deviations in structure or topology of cells. The type of extracted feature will then depend on the level of analysis chosen by the conductors of the engine design. To measure the deviations at the cellular level, morphological, textural, fractal, and/or intensity-based features can be used. In extracting such kinds of features at the cellular-level, the exact locations of the cells should be determined beforehand. On the other hand, to measure the changes at the tissue level, textural, fractal, and/or topological features can be extracted. ([8]) Most of automated diagnosis engines rely on the intensity values of the pixels and less on the topology information of the cells. However, intensity values are remarkably sensitive to noise in images so image pre-processing has to be performed before any feature extraction (as it's the case with our multiple cell segmentation pipelines).
For the enumeration and description of the different features, our work is inspired by the systematic survey of Cigdem Demir and Bulent Yener ([8]):

- **Morphological features:** morphological features give information about the size and the shape of cells. Amongst them, we can mention area, perimeter, radius, compactness, roundness, smoothness, length of minor and major axes, symmetry and concavity. These properties are generally sufficient to differentiate normal cells from malignant ones. Moreover, they allow some computation that can lead with some success to cancer detection in a specific tissue.

- **Textural features:** texture is a connected set of pixels that occurs repeatedly in an image. It provides information about the variation in the intensity of a surface by quantifying properties such as smoothness, coarseness, and regularity. The two most commonly used models to describe such features are the co-occurrence matrix and the run-length matrix. As it is not the purpose of this work to go deep into the mechanics of those, we won't go into further details (but we can redirect the reader to the work of R.M. Haralick ([56]) and M.M. Galloway ([57]). In fact, most of the description model for textural features involves intensity value matrices and it has been the case since the beginnings of automated image analysis processes: back in 1973, Haralick already uses grey-tone spatial dependence matrices in order to study such features ([58]). There are also other methods to extract textural features. One of such methods uses wavelets representation which discriminates several spatial orientations in the image ([59]).

- **Fractal-based features:** a fractal is an object with a self-similarity property, i.e., it appears the same at different magnifications [60]. The fractal geometry provides information on the regularity and complexity of an object by quantifying its self-similarity level. For that, the fractal dimension of the object is computed; unlike the Euclidean geometry, this dimension can be fractional. Fractal analysis is used to understand different phenomena in different biomedical applications including the cancer diagnosis [61], [62]. For the fractal analysis of a cell or a tissue, the most common feature is the fractal dimension [63], [64], [65], [66], [67]. In addition to the fractal dimension, the lacunarity is used for the purpose of fractal analysis of a cell or a tissue [68], [42]. The lacunarity quantifies the deviation from homogeneity in the texture. It is measured by computing the size of the holes on the fractal, i.e., counting the white pixels in the fractal image.

- **Topological features:** basically, topological features allow to study the structure of objects into bigger agglomerates. For the purpose of cell detection, it aims at analysing the spatial interdependency of the cells into the tissue of interest. There are several tools to achieve this topological analysis purpose but the two most common ones are Voronoï diagrams - with their dual graph, the Delaunay triangulation - ([18], [19], [22]) and (cell-)graphs ([69],[20],[21],[23]). The Voronoï diagrams and Delaunay graphs are geometrical models that can be used for quantifying the spatial distribution of objects in one image. In this work, these objects are cells. They give information about the spatial distribution of objects but also the geometrical relationships between them and, this was mentioned previously, this kind of information is critical if we want to get relevant topological information about the cells into one analysed tissue. More specifically, we discuss here what those models are as we use them for our own multi-approach feature extraction process.

  - *Voronoï diagrams*

    Voronoï diagram for one image I and a set of points P belonging to I is the result of determining areas of points belonging to the image which are closer to one point of P than to the others. Here's a more formal definition ([20]):

    A single Voronoï polygon, V_x is defined by the halfplanes $H(p_x p_y)$ that perpendicularly bisect the lines between a centre point and its neighbouring *point-like seeds. When we use this rule for every point considered, the area of interest is completely covered by adjacent polygons, constituting the Voronoï Diagram (VD).*

Figure 25 shows an example of a Voronoï Diagram obtained from a processed H&E stained prostate image.

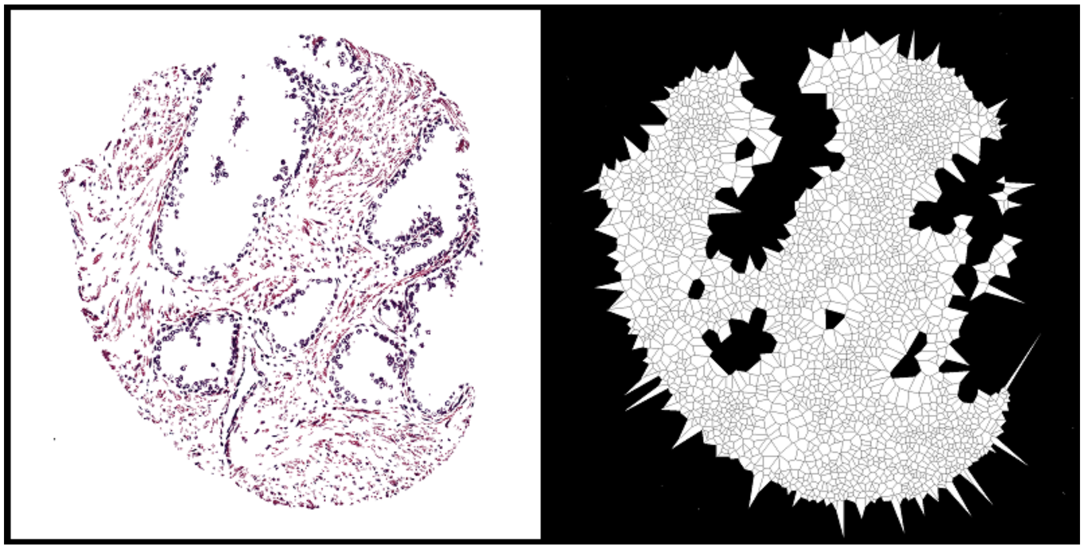


Figure 25 - Voronoï diagram of a processed H&E stained prostate image (marginal polygons have been ignored)

From the use of the Voronoï diagrams, we can extract several features such as:

- **Average, median and standard deviation of the Voronoï polygons**: $area_{avg}, area_{med}, area_{sd}$.
- **Area disorder**: $area_{dis} = \frac{1}{1+\frac{area_{sd}}{area_{avg}}}$. This feature reflects the homogeneity factor between the sizes of the Voronoï polygons ([20]).
- **Roundness factor:**
  Formula: $F_i = \frac{4\pi A_i}{L_i^2}$, where i is the number assigned to a Voronoï polygon, A_i and L_i the area and the perimeter of this polygon.

Average: $RF_{avg} = \frac{1}{N}\sum_{i=1}^{N} RF_i$ where N is the amount of Voronoï polygons.

Homogeneity: $RFH = \frac{1}{1+\frac{RF_{sd}}{RF_{avg}}}$

These features inform about the roundness of each Voronoï polygon and the homogeneity or the variation of this factor. They are generally crossed with the firstly introduced features for Voronoï diagram study ([75]).

- **Density:** $Dens = \frac{amount\ of\ Voronoi\ polygons}{total\ image\ area}$. This feature allows to observe how dense is the population of the Voronoï polygons within the image (marginal polygons, which are on the border of the image, are ignored) ([20]).

➢ *Delaunay triangulation*

Delaunay graph is actually the dual graph of a Voronoï diagram. Such a graph can be built by drawing a segment between each pair of points for which the Voronoï polygons are adjacent (separated by a Voronoï edge) ([76]).

Figure 26 shows an example of a Delaunay Graph obtained from a processed H&E stained prostate image.

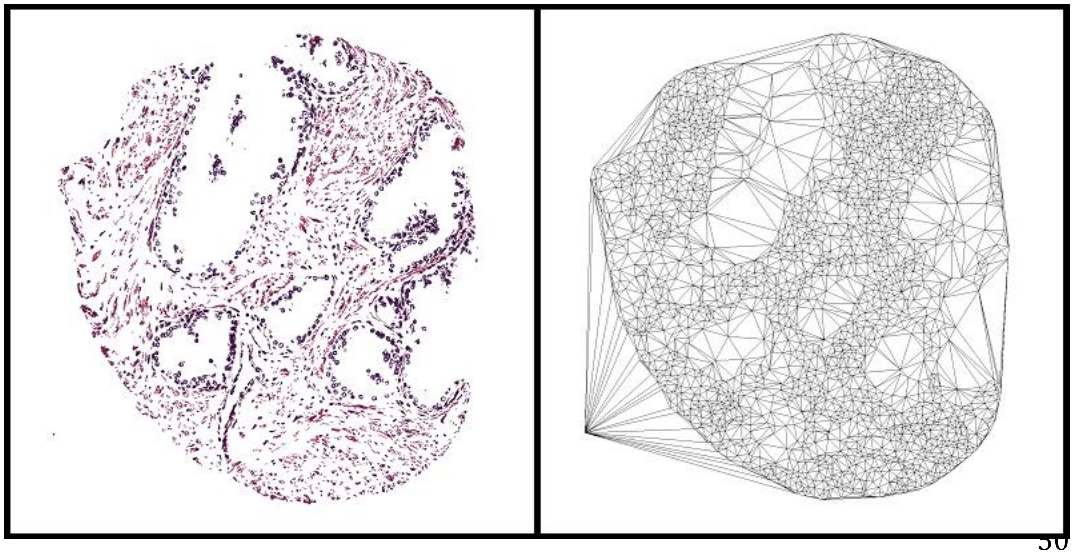


Figure 26 - Delaunay graph of a processed H&E stained prostate image

From the use of the Delaunay graphs, we can extract features such as:

- **Average and standard deviation of segment length:** $segment_{avg}, segment_{sd}$.
- **Segment length disorder:** $segment_{dis} = \frac{1}{1+\frac{segment_{sd}}{segment_{avg}}}$ ([20]).

➢ *Graph-based features*

Multiple studies have already paved the way for the use of graph theory in automated cancer engines ([21], [22], and [24]). Two of the major advantages of graph theory in this specific context are that the computational cost of algorithms working with graphs is often lower than other feature analysis approaches and that this field of knowledge is mature enough to allow all sorts of optimizations to increase performance and accuracy in multiple, different, situations. There are a lot of types of graph that are used for the purpose of cell topology analysis but the main idea is to use the cell locations, build a specific type of graph based on those locations and then to exploit the properties of this specific type of graph to try and highlight the ones that can distinguish different types of topologies or cell architectures. Those techniques indeed rely on the fact that cells are more chaotically organized the more a cancer is developed.

- **Intensity-based features:**

Some approaches to cell or tissue analysis rely on the colour properties of those. The main idea behind these approaches is that different objects react differently with the dyes and it’s particularly the case with normal and cancerous cells or tissues. Thus, the approaches compute some colour intensity features, either per colour channel or more generally, highlighting links through colour channel or using statistical measures for colour features in an image.

b) Feature selection

Once we have extracted several features from the analysed images. We could want to design and run a learning process that will allow to classify new images based our training set data. However, it is common to reduce the variable dimensionality in order to improve performance when predicting (with machine learning algorithms). Feature (or variable) selection can actually serve three purposes: *improving the prediction performance of the [automated] predictors, providing faster and more cost-effective predictors, and providing a better understanding of the underlying process that generated the data* ([78]). Those are exactly the same goals that we want to reach once we have generated lots of multi-variable data after analysing the images in the previous computing step of our decision making tool. Indeed we want to get rid of irrelevant features and keep only the ones that helps with a final verdict for the image classification, we also want our learning process to be as performant as possible since it will face a high volume of data and, finally, understanding which key features can be lead to a good prognostic or diagnosis is fundamental to improve our processes.

Many methods for feature selection are nowadays available and there often can be used in many different situations: information gain, chi-square, symmetrical uncertainty, reliefF, SVM-RFE, PCA (Principal Component Analysis), etc. ([80])

We didn't often find trace of those kind of algorithms in similar cancer diagnosis studies. That could be explain by the fact that the number of identified features is not so high and thus, such algorithms wouldn't be of great help. However, those feature selection algorithms could be used to actually ensure that each extracted feature can lead to a valuable verdict and, of course, once the amount of features gets high, it becomes an absolute necessity.

Finally, feature selection algorithms can solve another common problem introduced by considering irrelevant features, this problem is called *overfitting*. Overfitting is a well-known problem when designing classifiers or other statistical model and can lead to incorrect prediction or biased results by focusing on specificities of the training data instead of reflecting the underlying relationship between the extracted features and the prediction subject. *For example, in the domain of medical diagnosis, our purpose is to infer the relationship between the symptoms and their corresponding diagnosis. If by mistake we include the patient ID number as one input feature, an over-tuned machine learning process may come to the conclusion that the illness is determined by the ID number* ([79]).

B. Custom extracted and selected features

As introduced earlier one of our main objectives is to provide several tools for the design of a decision making assistant for the prognosis and diagnosis of the prostate carcinoma. More specifically we want those tools to be free, open and available for further improvement and refinement. We also tried our best to provide different tools with a multi-approach view, which is characterized by the consideration of multiple analysis and feature extraction/selection methods and processes.

With this idea in mind we performed a wide analysis with 5 feature types and 35 single features of each image in our initial image set.

a) Morphological features

For this first type of features, we selected the most common features available: size, perimeter and shape.

For each cell object recorded during the image processing phase we computed the mean and the standard deviation for these three features.

As the shape feature is concerned, we used the ratio between perimeter and size and the centroid. The first information indicates how stretched an object is and the second one (g) is obtained as follows:

$$\begin{cases} g_x = \sum_{i=0}^{N} x_i \\ g_y = \sum_{i=0}^{N} y_i \end{cases}$$ With N being the amount of points in the shape.

b) Colour intensity features

For the colour analysis of the images, we extracted the mean and the standard deviation for every RGB colour channel. We also extracted the start and end values of a specific interval of values within the colour channel. This interval has been built around the maximum value in the frequency array of the colour channel and increased by looking around this value for the biggest value frequency. That way, it would give us a more precise idea of the distribution of the colour channel values in the image.

Figures 27, 28 and 29 shows the colour channel distribution for an H&E stained image with removed background as well as this *percentile interval*. For this example, and for our tests, we used a 75 percentile. The choice for this value has been guided by the will to get rid of *noise values* in the images (values are anecdotally present in the image). The percentile interval is represented in the three next figures with dark grey vertical lines.

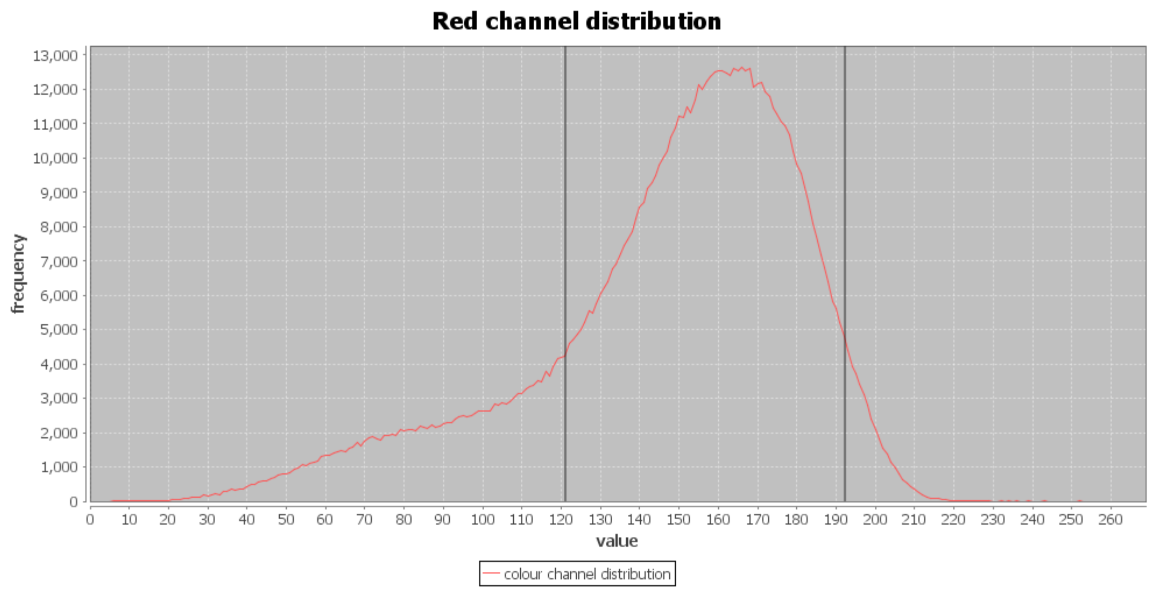


Figure 27 - Red colour channel distribution for an H&E stained image with removed background

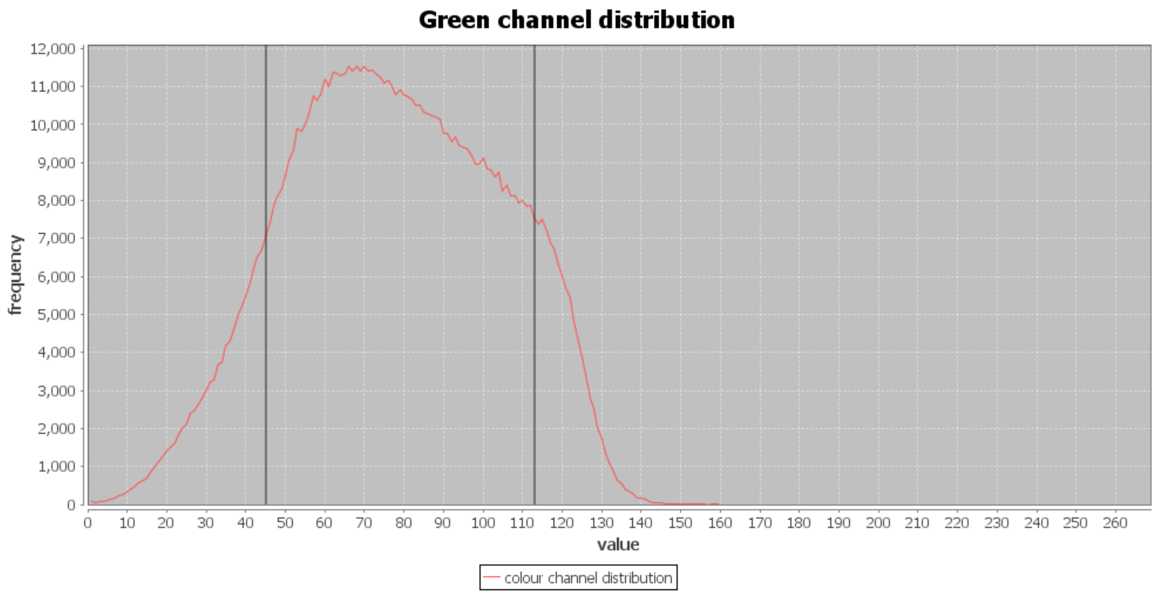


Figure 28 - Green colour channel distribution for an H&E stained image with removed background

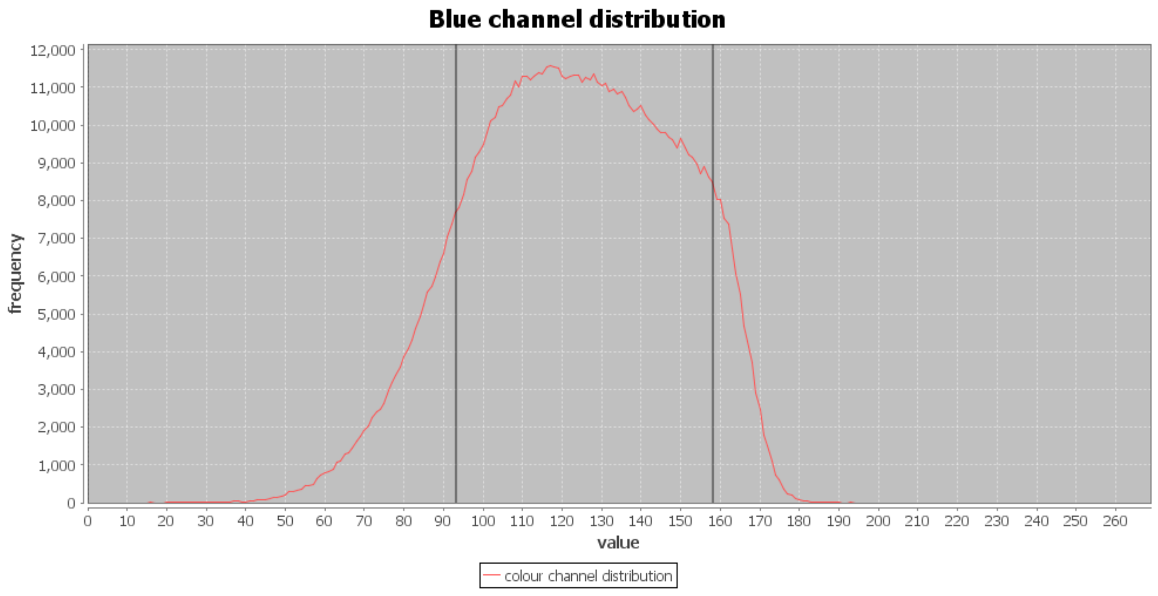


Figure 29 - Blue colour channel distribution for an H&E stained image with removed background

```
PERCENTILE INTERVAL ALGORITHM (JAVA-LIKE CODE)

BLUE COLOUR CHANNEL CASE

/**

 * This method computes the X percentile interval for the

 * blue colour channel.

 *

 * X: Desired percentile.

 */

int[] getInterval(float X){

   // blue array is the frequency array for the blue colour channel

   int idx_blue_max = getMaxBlueFrequencyIdx();
   int idx_blue_left = idx_blue_max - 1;
   int idx_blue_right = idx_blue_max + 1;
   int totalX = Math.round( total * X );
   int tmpBlue = blue[idx_blue_max];

   while ( tmpBlue < totalX ) {
      if ( blue[idx_blue_left] > blue[idx_blue_right] ) {
          tmpBlue += blue[idx_blue_left];
          idx_blue_left--;
      }
      else {
         tmpBlue += blue[idx_blue_right];
         idx_blue_right++;
      }
   }

   intervalXBlue[0] = idx_blue_left;
   intervalXBlue[1] = idx_blue_right;

   return intervalXBlue;

}
```

c) Voronoï features

The next two types of features we extracted for image analysis are topological ones. We actually used those two dual graphs: Voronoï diagram and Delaunay graph. In this section we develop how we used the first geometrical construct for our analysis.

For the implementation of the Voronoï process, I relied on one of the most performant algorithm available for the construction of such graph. This algorithm is the Fortune algorithm, introduced by Steven Fortune in 1986 ([81]). This algorithm runs with a $\mathcal{O}(n.\log(n))$ time complexity and a $\mathcal{O}(n)$ space complexity, n being the amount of input sites (points) from which to build the Voronoï diagram.

As many implementations have already been developed in several programming languages for this algorithm, we decided to reuse a reliable implementation available in Java by Ian Cameron Smith. He relied himself on a C# implementation of the algorithm developed by Benjamin Dittes.

Once again, as it isn't the purpose of this work, we won't go into any more detail about the mechanisms involved for the design of the Fortune algorithm (the reader can read about that in the original paper of Fortune himself in [81]) but we will concentrate on the modifications and optimizations we performed to adapt the algorithm to our purpose.

The first modification we wanted to introduce is to allow the implementation to actually draw the Voronoï polygons resulting from the application of the Voronoï process on an H&E stained image. As the result of this original implementation was a graph, we had to add the drawing part that relies on the edges of the graph.

Moreover the implementation we reused lacked of the marginal polygons suppression optimisation introduced in [20]. This optimization tends to eliminate the Voronoï polygons resulting from the border effect in the construction of the graph. Such border polygons don't reflect the structure of the studied object and can actually bias the results of further analysis of the Voronoï diagram.

The way we implemented this last optimization is rather simple and relies on two conditions for the filtering of the polygons:

- If the polygon is at the border of the image, it's rejected.
- We record all the polygon sizes for the current Voronoï process. Then we sort the resulting list of sizes and we set a size threshold based on a determined percentage of the size distribution (this percentage value has actually been computed from several observations on multiple images but can, of course, be configured as an input of the algorithm).
  Note that this size threshold setting process could be improved thanks to a *Lagrange* or *spline* interpolation of the size distribution. Thanks to those, we could make the threshold dynamically vary for each case of analysis thanks to the analysis of the derivative of such polynomial expressions. Rudiments of such processing can be found in our open source code but is open to further development.

Figure 30 shows the results of the Voronoï process on an H&E stained image without the marginal polygons suppression optimisation whereas 31 shows the same process on the same image but with this optimisation.

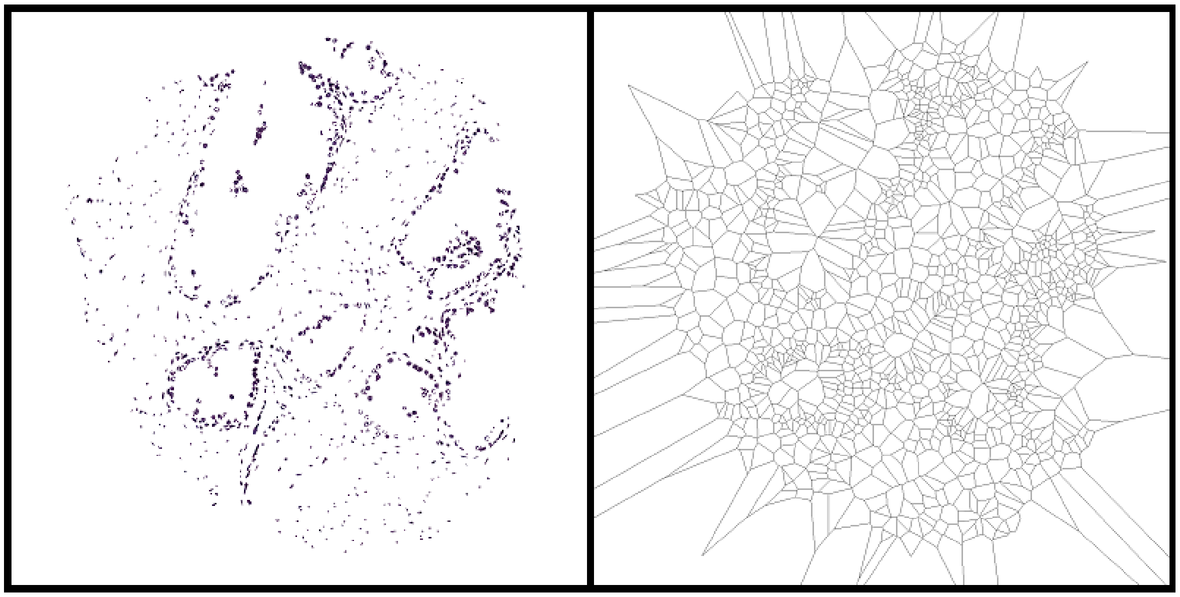


Figure 30 - Result of the Fortune algorithm for Voronoï diagram without optimization

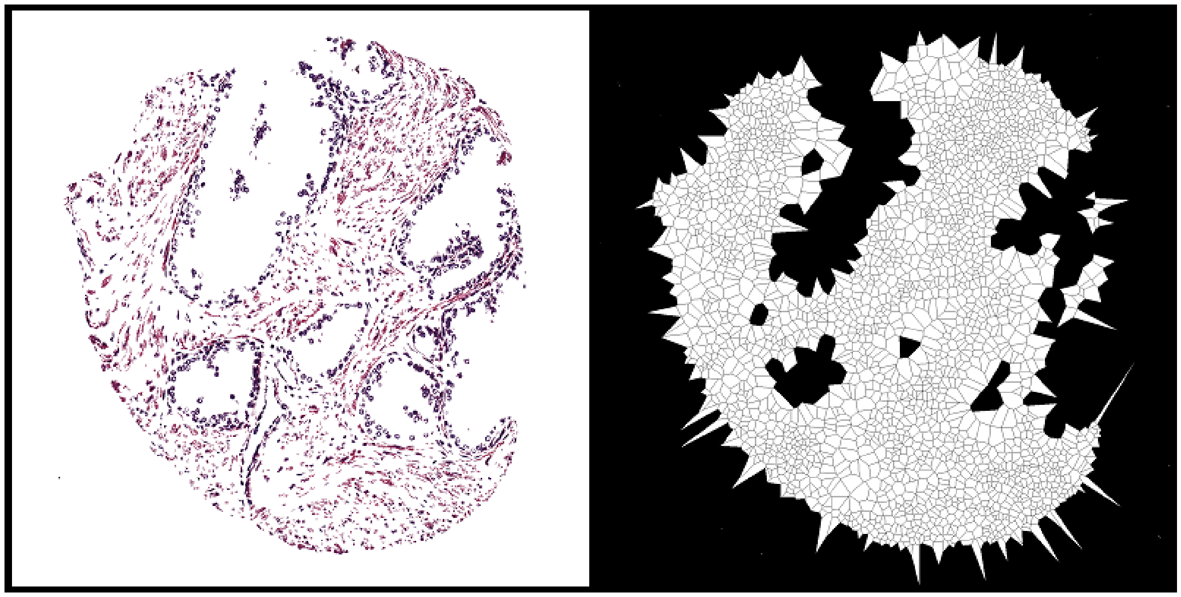


Figure 31 - Result of the Fortune algorithm for Voronoï diagram with the border effect optimisation

The reason why we introduced the size threshold is that *border polygons* are often larger than other polygons. We then study the size charts for the size of the polygons in multiple Voronoï diagrams (resulted from the analysis of H&E stained images) and select a sensible threshold based on our observations.

Figure 32 shows a common size distribution for the Voronoï polygons and how we can select the size threshold.

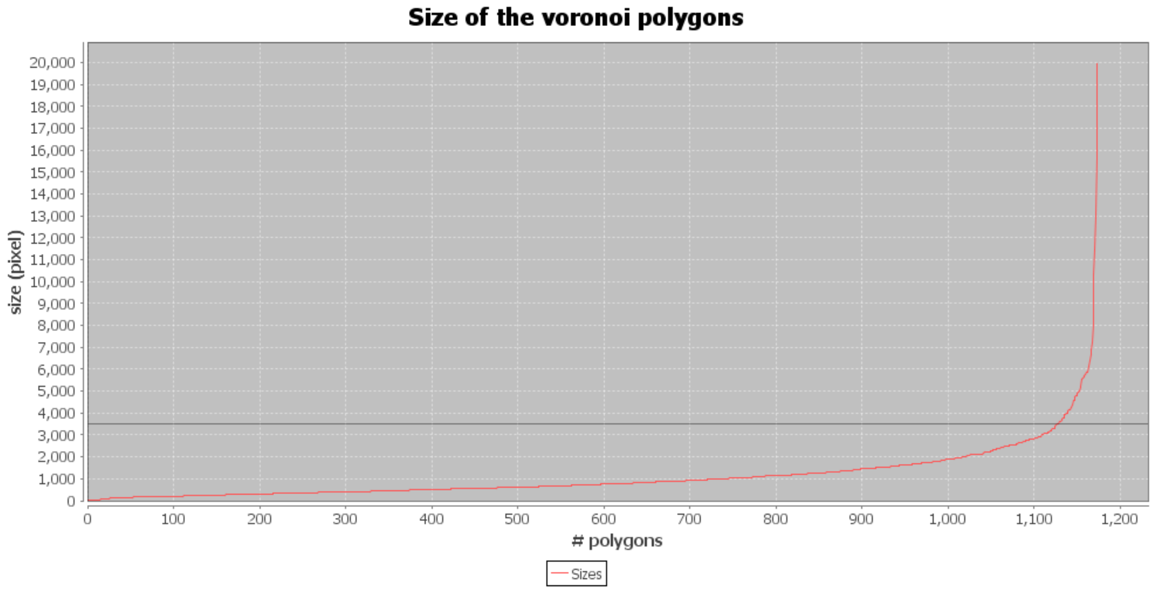


Figure 32 - Voronoï polygons size distribution with static threshold (4% of the distribution is discarded in this example)

As for the analysis of the produced Voronoï diagrams, we extracted 7 features:

- Mean value, the median, and standard deviation for the area of the polygons.
- Area disorder:
  The Area denotes the area of a single Voronoï polygon, measured in pixels. Area disorder reflects the variation in the polygons associated with considered population of point-like seeds. The feature acquires the value of 0 if all polygons have the same area and tends towards 1 otherwise. The entire population except the marginal polygons is considered ([20]).

$$A_{dis} = 1 - \frac{1}{1 + \frac{area_{stdDev}}{area_{avg}}}$$

- Average roundness factor:
  This equals the roundness factor of one polygon average RF is the average over the entire population except for the marginal polygons ([20]).

$$RF_{avg} = \frac{4\pi}{perimeter^2}$$

- Roundness factor disorder:
  This features acquires the value of 1 if all the RF's are the same and tends towards zero otherwise. The entire population except the marginal polygons were considered ([20]).

$$RF_{dis} = 1 - \frac{1}{1 + \frac{RF_{stdDev}}{RF_{avg}}}$$

- Density:
  This feature represents the density of the entire population, except for the marginal polygons, which are eliminated ([20]).

$$d = \frac{\#\, polygons}{\sum area}$$

d) Delaunay features

Voronoï diagrams are commonly used with their dual graph, which are Delaunay graphs. In this section we develop how we used those graphs in order to extract features from the studied H&E stained images.

As the design of Delaunay graphs is concerned, we decided to re-use an existing Java implementation by X. Philippeau which has been developed on basis of the algorithm introduced by Guibas and Stolfi in [81]. This implementation is both accurate and performant since it has the same time and space complexities as the ones introduced for the construction of the Voronoï diagram in the previous section (respectively, $\mathcal{O}(n.\log(n))$ and $\mathcal{O}(n)$).

Apart from the adaptation of the original source code to the needs of our analysis tool (full integration and drawing), we didn't perform any other modification or optimization to the Java source code.

Figure 33 shows the result of the Delaunay graph generation process applied to an H&E stained prostate image.

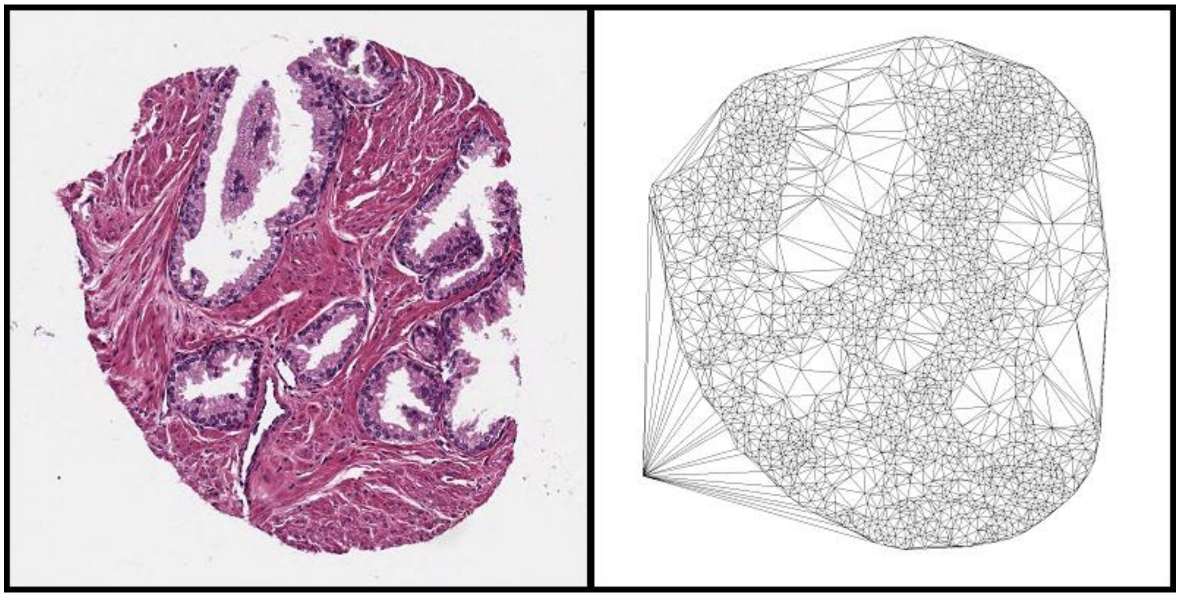


Figure 33 - Delaunay graph generation process applied to an H&E stained prostate image (the image has been previously processed by one of our pipelines)

One of the major improvements that could be performed for this Delaunay graph generation process is to remove, as it has been implemented for the Voronoï diagram generation process, the marginal Delaunay segments resulting from the well-known border effect. Unfortunately, we didn't have time to implement the counterpart optimization for this particular process but the source code is obviously open to further development.

Now we have introduced the design of the Delaunay graph generation process let us go over the features we extracted thanks to it. We actually extracted 3 features:

- Mean value and standard deviation for the length of Delaunay segments;
- Delaunay segment length disorder:

$$DL_{dis} = 1 + \frac{1}{1 + \frac{DL_{sd}}{DL_{avg}}}$$

The reason why we didn't choose to extract more features from this kind of graph is that the selected ones don't require intensive computational processes, which is often the case with other types of features ([20]). One improvement to this approach could be to extract more features and then select the ones that truly have an impact at the end of the learning process (see Feature Selection section). This would be a good compromise between accuracy and performance issues.

e) Graph features

The last type of features we used for our image analysis is another topological information about the cell distribution in an H&E stained prostate image. We decided to introduce at least one graph feature type in our analysis because it has proven to be quite effective for the learning process of such prognosis or diagnosis tool for cancer in some other studies and because it seems to be one of the more recent approaches to image analysis in various automated cancer diagnosis tools ([21], [22], [24], [83], [84]).

Our choice for the graph feature type has been to work with the Waxman random graph model. Let us introduce some rudiments about this model (we partly rely on explanations in [85] and [86]):

The Waxman graph is a random graph model. It considers a distribution of N nodes in an x-y-coordinate plane from which an edge probability will be computed for each pair of node u and v.

Let $d(u,v)$be the Euclidean distance between those two nodes, L the maximum Euclidean distance between two nodes in the node set and parameters α and β set by the user. The probability that an edge is drawn between u and v is computed as follows:

$$P_{edge}(u,v) = \beta^{-\frac{d(u,v)}{L\alpha}}$$

α and β are actually two parameters set between [0,1] that will impact the probability of an edge to be added in the final Waxman graph. *Larger values of β result in graphs with higher edge densities, while small values of α increase the density of short edges relative to longer ones* ([86]).

As we couldn't find satisfactory implementations of such graph model in Java we decided to implement the model ourselves based on the following code basis:

Waxman basis algorithm (Java-like code)

```
boolean edges[][] = new boolean[points.size()][points.size()];

double e = Math.E;

for ( int i = 0; i < points.size(); i++ ) {

   for ( int j = 0; j < points.size(); j++ ) {

      probabilities[i][j] = beta * Math.pow( e, ( 0 - distances[i][j]
) / ( alpha * maxDistance ) );

      if ( probabilities[i][j] > new Random().nextDouble() &
distances[i][j] != 0 ) {

         edges[i][j] = true;

      }

   }

}
```

Figure 34 shows the result of the Waxman graph generation process on an H&E stained prostate image.

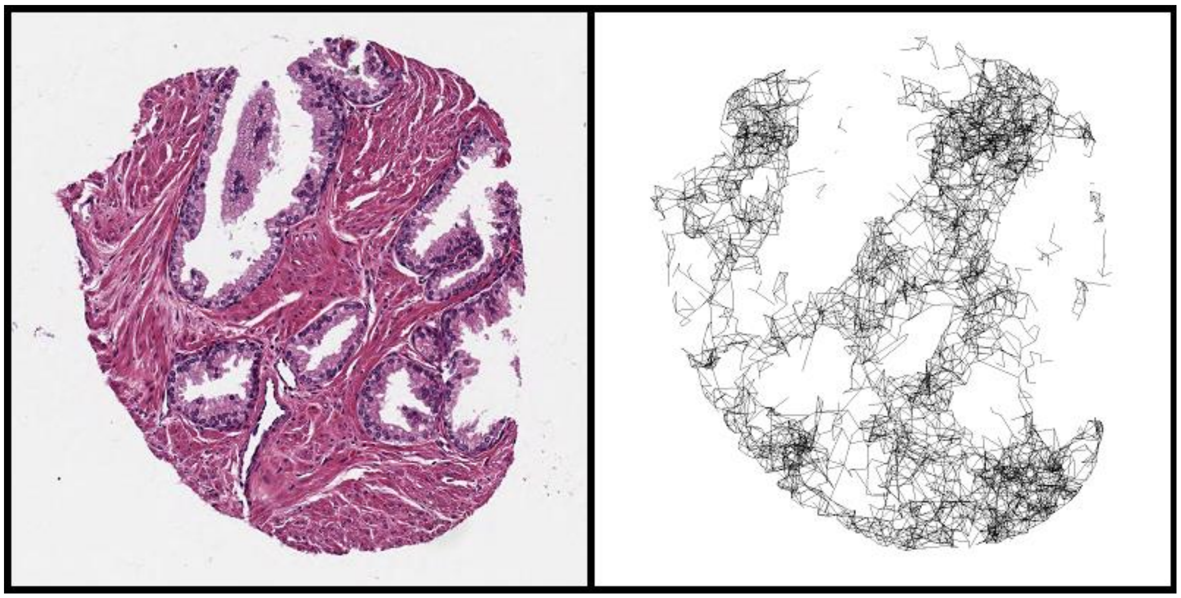


Figure 34 - Waxman generation process applied to an H&E stained prostate image. Alpha=0.1, Beta=1.

After the Waxman graph generation we extract 6 features from the resulting graph:

- Mean value and standard deviation for the degree of the nodes;
- Mean value and standard deviation for the weighted degree of the nodes (computed thanks to the distance weight assigned to the edges);
- Mean value and standard deviation for the clustering coefficients.

  The clustering coefficient C is defined as follows. *Suppose that a vertex v has kv neighbours; then at most $\frac{k_v(k_v-1)}{2}$ edges can exist between them (this occurs when every neighbour of v is connected to every other neighbour of v). Let C_v denote the fraction of these allowable edges that actually exist. Define C as the average of C_v over all v* ([87]).

Further improvements regarding the graph features would include more features for the Waxman graph model (such as eccentricity or closeness) and eventually more type of graphs in order to see which one is more suitable in a particular situation.

f) Analysis statistics

This phase of feature extraction or image analysis has been performed on our dataset for the first three image processing pipelines introduced in the previous chapter. We couldn't run this analysis for the fourth pipeline as it would have required to select the seeds manually for around 300 H&E stained prostate images and we just lacked time for this to be performed.

Here's what we could observe on our standard configuration:

- Average running time for the analysis on pipe-1-processed images is 8.56 seconds.
- Average running time for the analysis on pipe-2-processed images is 9.30 seconds.
- Average running time for the analysis on pipe-3-processed images is 14.76 seconds.

Those results were actually expected as the pipes gradually leaves more points in the image resulting from the image processing stage. As the complexity of our analysis depends uniquely on this factor, it is then normal to see these growing running times.

Table 1 shows the total time for each of our three first pipes including both image processing and image analysis.

Table 1 - Computing times for the image processing and image analysis stages.

	Image processing time (s)	**Image analysis time (s)**	**Total computing time (s)**
Pipe-1-processed	2.02	8.56	10.58
Pipe-2-processed	6.99	9.30	16.29
Pipe-3-processed	0.56	14.76	15.32

It is quite interesting to see that the association of the first image processing pipeline with this raw analysis process (without feature selection) represents the best time performance whereas the third image processing pipeline was almost four times faster initially.

g) Selected features

Once we extracted the features from the images in our dataset we had to face the issue of either considering all of the features for the next step, which is machine learning, or to reduce the dimensionality of our processed dataset to improve the performance of the latter and remove features that wouldn't impact (at least significantly) the potential decision in terms of prognosis or diagnosis of submitted images to a final decision making tool designed thanks to our open-source tool suite.

The answer to this question was not as trivial as it could seem because if we completely understood the benefits of performing such feature selection process we weren't absolutely sure that our processes were optimally designed and implemented, or that they didn't have an improvement margin that could impact on the representativeness of specific features in the final decision.

As it was not an easy choice, we just decided to do both, that is to see what could be the results of performing a feature selection process on our dataset in its current stage and then to use also our dataset as it was for the next part of processing, which was machine learning.

As the choice for a feature selection algorithm was concerned, we decided to use the principal component analysis (PCA) which has proven to be one of the major and most reliable techniques for the purpose of dimensionality reduction. The choice for this particular algorithm was also motivated by the fact that it was one of the available algorithms in the *Weka* machine learning platform that we decided to use for all our operations related to machine learning. On top of that, with the same idea of not losing features maybe immaturely extracted, the principal component analysis has the advantage of feature selection without really performing it, the technique is actually a modification or improvement of a feature extraction process by expressing the extracted features in a way that reflects their weight in the classification (here, normal or malignant image).

As it has been previously the case for other algorithms in this work, we won't go into the deepest level of details to describe the technique but we want to make sure that the reader can acquire – if it's not already the case – a solid intuition about how and why we use principal component analysis. For this discussion, we partly rely on the work of Jon Shlens from the Princeton university ([88]): principal component analysis is a variable reduction (often assimilated to feature selection) algorithm. Its wide use is often explained by the simplicity and non-parameterisation of the method and the fact that it is powerful enough to extract *relevant information from confusing data sets. With minimal additional effort PCA provides a roadmap for how to reduce a complex data set to a lower dimension to reveal the sometimes hidden, simplified dynamics that often underlie it*.

The idea behind the functional mechanism of PCA is a simple hypothesis: linearity. Indeed when using PCA we make the superimposition that *the data characterizes or provides an ability to interpolate between the individual data points*. However it is to be noted that most complex systems are nonlinear, and it might actually be the case for our case study, and *their main qualitative features are a direct result of their nonlinearity*. This is actually partly an obstacle thanks to the fact that *locally linear approximations usually provide a good approximation because non-linear, higher order terms vanish at the limit of small perturbations*. That's exactly what we are aiming for at this stage of processing.

In general, each data sample is a vector in the m *dimensional space, where* m *is the number of measurement types [extracted features]. Equivalently, every data sample is a vector that lies in an* m*-dimensional vector space spanned by an orthonormal basis. All measurement vectors in this space are a linear combination of this set of unit length basis vectors. A naive and simple choice of a basis B is the identity matrix I*

$$B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & 1 \\ 0 & 0 & \cdots & 1 \end{bmatrix} = I$$

where each row is a basis vector b_i *with* m *components.*

In our particular case that would mean that each data sample corresponds to a thirty-five dimensional column vector $\vec{X}$

$$\vec{X} = \begin{bmatrix} morphological\_sizeAvg \\ intensity\_redAvg \\ voronoi\_areaDisorder \\ delaunay\_edgeStdDev \\ waxman\_clusteringCoefficient \\ \vdots \end{bmatrix}$$

where each measurement type contributes two a diagnosis (normal/malignant image) and the entire vector $\vec{X}$ is the set of coefficients in the naïve basis B.

Now we can finally see how PCA will proceed to perform the dimensionality reduction. Indeed it will basically look for a better basis, which is a linear combination of the initial basis that best re-expresses the dataset. The question will then be to find an appropriate basis transformation and the principal components will be the raw vectors in the transformation. Thus, more mathematically formed, we have:

Let X and Y be $m \times n$ matrices related by a linear transformation P. X is the original recorded data set and Y is a re-representation of that data set.

$$PX = Y$$

The rows of P are a set of new basis vectors for expressing the columns of X.

As previously mentioned before, we took benefit from the availability of a reliable PCA implementation in the *Weka* machine learning platform and we just had to run it with a basic pre-configuration on our three pre-processed datasets.

Table 2, 3 and 4 show the output of the PCA process in the *Weka* environment for pipe-1, pipe-2 and pipe-3-processed images respectively.

Table 2 - Output from Weka for the Principal Component Analysis on the pipe-1-processed dataset

0.6364	1	-0.258intensity_B90End-0.258intensity_G90End-0.253intensity_R90End-0.25intensity_GAvg-0.241intensity_BAvg-0.239intensity_RStdDev...
0.4606	2	0.289graph_degreeStdDev+0.285graph_degreeAverage+0.283graph_weightedDegreeAverage+0.266graph_weightedDegreeStdDev+0.239intensity_R90Start+0.226intensity_B90Start...
0.3059	3	-0.378Voronoï_sd-0.336delaunay_edgeStdDev-0.315Voronoï_mean-0.298delaunay_edgeAverage+0.297Voronoï_density-0.275delaunay_lengthDisorder...
0.2326	4	0.38 morphological_shapeStdDev+0.362intensity_BStdDev+0.36 morphological_shapeAvg+0.244intensity_RStdDev+0.236intensity_GStdDev+0.227graph_weightedDegreeStdDev...
0.1753	5	0.675graph_clusteringCoefficientStdDev+0.667graph_clusteringCoefficientAverage-0.121morphological_perimeterAvg-0.119morphological_sizeAvg-0.116morphological_perimeterStdDev-0.114morphological_sizeStdDev...
0.1354	6	-0.481morphological_shapeStdDev-0.389morphological_perimeterStdDev-0.349morphological_sizeStdDev-0.309morphological_perimeterAvg-0.251morphological_sizeAvg-0.228morphological_shapeAvg...
0.0988	7	0.642Voronoï_averageRoundnessFactor+0.585Voronoï_roundnessFactorDisorder+0.227Voronoï_areaDisorder+0.158graph_weightedDegreeStdDev-0.147Voronoï_median+0.141graph_degreeStdDev...
0.0734	8	-0.413Voronoï_areaDisorder+0.38 Voronoï_median-0.34delaunay_lengthDisorder+0.314Voronoï_averageRoundnessFactor+0.301graph_weightedDegreeStdDev+0.248Voronoï_mean...
0.0574	9	0.626morphological_sizeStdDev-0.461morphological_shapeStdDev-0.309morphological_perimeterAvg+0.234intensity_BStdDev-0.221morphological_sizeAvg+0.192morphological_perimeterStdDev...
0.0418	10	0.428intensity_GStdDev-0.383morphological_shapeAvg+0.368intensity_BStdDev-0.328morphological_shapeStdDev+0.279Voronoï_density-0.249Voronoï_roundnessFactorDisorder...

Table 3 - Output from Weka for the Principal Component Analysis on the pipe-2-processed dataset

0.7589	1	0.321intensity_BAvg+0.321intensity_GAvg+0.315intensity_B90Start+0.31 intensity_G90Start+0.306intensity_G90End...
0.5452	2	0.326delaunay_edgeAverage+0.322Voronoï_mean+0.313Voronoï_median-0.308Voronoï_density+0.302delaunay_edgeStdDev...
0.3887	3	-0.344intensity_RAvg-0.336intensity_R90Start+0.317intensity_GStdDev+0.242intensity_BStdDev-0.241delaunay_lengthDisorder...
0.29	4	-0.367morphological_perimeterAvg-0.366morphological_perimeterStdDev-0.36morphological_sizeAvg-0.353morphological_sizeStdDev-0.221intensity_B90End...
0.2055	5	0.296delaunay_lengthDisorder-0.287intensity_RAvg+0.276intensity_GStdDev+0.263intensity_BStdDev-0.254morphological_perimeterStdDev...
0.1503	6	0.687graph_clusteringCoefficientAverage+0.68 graph_clusteringCoefficientStdDev+0.115morphological_sizeStdDev+0.112morphological_perimeterStdDev+0.071intensity_BStdDev...
0.1133	7	0.461Voronoï_averageRoundnessFactor+0.389Voronoï_areaDisorder+0.387Voronoï_roundnessFactorDisorder-0.343Voronoï_median-0.249morphological_shapeStdDev...
0.0898	8	0.791morphological_shapeStdDev-0.271intensity_BStdDev+0.186delaunay_lengthDisorder+0.186Voronoï_areaDisorder-0.157graph_weightedDegreeStdDev...
0.0676	9	-0.45Voronoï_roundnessFactorDisorder-0.446morphological_shapeAvg-0.433Voronoï_averageRoundnessFactor-0.268intensity_BStdDev-0.264morphological_shapeStdDev...
0.0502	10	0.399intensity_BStdDev-0.382graph_weightedDegreeStdDev+0.327morphological_shapeAvg+0.292Voronoï_areaDisorder-0.289graph_degreeStdDev...
0.0382	11	-0.618morphological_shapeAvg+0.299morphological_shapeStdDev+0.269intensity_BStdDev+0.267Voronoï_averageRoundnessFactor+0.259intensity_R90Start...

Table 4 - Output from Weka for the Principal Component Analysis on the pipe-3-processed dataset

0.6319	1	-0.259intensity_G90End-0.252intensity_B90End-0.249intensity_R90End-0.248intensity_GAvg-0.239intensity_BAvg...
0.4341	2	-0.31intensity_BStdDev-0.301intensity_RStdDev-0.292intensity_GStdDev+0.219intensity_R90Start+0.213intensity_RAvg...
0.3148	3	-0.409morphological_perimeterAvg-0.389morphological_sizeAvg-0.364morphological_perimeterStdDev-0.342morphological_sizeStdDev+0.228Voronoï_sd...
0.2334	4	0.444delaunay_lengthDisorder+0.43 Voronoï_areaDisorder+0.313Voronoï_sd+0.28 morphological_sizeStdDev+0.276morphological_perimeterStdDev...
0.1762	5	0.661graph_clusteringCoefficientAverage+0.651graph_clusteringCoefficientStdDev-0.237Voronoï_averageRoundnessFactor+0.108morphological_shapeStdDev-0.093graph_weightedDegreeStdDev...
0.1304	6	0.356Voronoï_averageRoundnessFactor+0.335Voronoï_median+0.302morphological_perimeterStdDev+0.292morphological_sizeStdDev+0.291Voronoï_mean...
0.0953	7	-0.599Voronoï_averageRoundnessFactor-0.313Voronoï_roundnessFactorDisorder+0.217intensity_GStdDev+0.216intensity_RStdDev+0.193intensity_B90End...
0.0723	8	-0.396Voronoï_areaDisorder-0.392morphological_shapeAvg-0.236intensity_RStdDev-0.234morphological_perimeterAvg-0.23intensity_GStdDev...
0.0531	9	0.71 morphological_shapeAvg+0.335graph_weightedDegreeStdDev+0.294graph_degreeStdDev-0.284Voronoï_density-0.208Voronoï_sd...
0.0389	10	0.638Voronoï_roundnessFactorDisorder+0.372morphological_shapeStdDev-0.304morphological_perimeterAvg-0.242Voronoï_density+0.23 morphological_sizeStdDev...

The key to read those tables is the following: PCA ranks the linear combination of feature types by their variance. That means that the first linear combination of feature types has a larger variance than the eighth one for example. Moreover that means that the first linear combination will be the one for which the values are the most widely distributed whereas the last combinations take their values in a small interval.

From the obtained results with PCA we can also see how the processing affected the way feature types are represented in the different datasets and this is quite a valuable result. Indeed we can see that if it is clear that the combinations involving intensity feature types are the most spread around their mean value for the three types of image processing, on the other hand we can observe that the first image processing pipeline will have the Waxman feature types widely spread, the second one the Delaunay/Voronoï feature types, and the third one, once again, the intensity feature types.

As for the valuable asset that PCA represents for the following stage of processing, machine learning, we decide to go over this subject in the next and last main chapter in this work.

5. Machine learning

In this last main section we introduce past work which consisted in including machine learning in cancer diagnosis tools and how we decided to use those techniques in our own custom diagnosis tool suite.

A. State of the art

When it comes to machine learning we can basically distinguish two ways of coping with the building of knowledge. We can either opt for unsupervised or supervised learning techniques.

As introduced in our initial overview unsupervised learning can be defined as follows ([15]): *unsupervised learning studies how systems can learn to represent particular input patterns in a way that reflects the statistical structure of the overall collection of input patterns*.

This kind of process can be particularly useful when we basically don't know what we are actually looking for when it comes to analyse a large input dataset. In case there is no rule, no criteria, no benchmark or even no information about the content of this dataset unsupervised learning may be the right method to recognise patterns and therefore pave the way for further rule establishment in a more structured learning process.

This is exactly the kind of situation we have to face when designing an automated prognosis or diagnosis tool suite after the image processing and feature extraction/selection stages. We actually don't know what we are looking for: we don't really know how to distinguish normal cells from cell aggregates and we don't know which extracted feature value ranges can be assigned to each particular group of image objects. More than that, we don't even know how which groups to make between the image objects.

To overcome such problem one idea would be to gain information from the already pre-processed data we obtained with the first tools of our suite thanks to unsupervised learning.

As it is not our goal to provide and comment many state-of-the-art techniques or methods for unsupervised learning, we won't be talking about *factor analysis, PCA* (this method has been described earlier in this work), *mixtures of Gaussians, ICA, hidden Markov models, state-space models, EM algorithms, Bayesian inference, Laplace approximation, BIC* (Bayesian Information Criterion)*, variational approximations, and expectation propagation (EP)* ([89]) but we will focus on probably the most widely used and the one that we actually used as a basis for one of our tools, the clustering technique.

a) Clustering

Clustering is actually a technique used in very various contexts ranging from feature construction ([13]) to actual knowledge acquisition. We even used the technique for our second image processing pipeline and it is quite common to use clustering for sorting, classification or labelling of little-known or highly multi-dimensional items in a given collection.

It is also quite common to use clustering algorithms for various purposes in automated diagnosis engines ([21], [22], [24], [90], [91], [92], [93] and many more) and that's actually one reason that guided us into designing a processing tool in our diagnosis engine suite.

Basically if we had to summarize the main principle behind the wide concept of clustering we would say that it is aimed to group items in a collection, giving a meaning to those defined groups. One of most critical sub-concepts used in clustering algorithms is obviously *similarity*. Each clustering algorithm will define its own definition or representation of similarity. For example, a basic *k-means* clustering algorithm would define the similarity factor by using the Euclidean distance as a way to group items and make clusters out of a raw collection. By *recognizing patterns of similarity* between objects clustering allows to classify, compress or even explore the underlying structures that are at the origin of a data collection. In the context of an automated diagnosis engine that would mean that it could help us to determine which imaging feature set has a representative impact on the final diagnosis decision, to understand maybe more how tissues and cells could or should be analysed in order to produce a good estimation for a prognosis or a diagnosis.

Data clustering well often comes with several issues that question the validity or even the purpose of such technique and those issues have to be addressed when designing a clustering tool for data labelling, compression or classification ([96]):

- How can we define a cluster?
- How can we choose a similarity metric?
- Which features can we work with? How can we normalize them?
- Which is the most appropriate data representation for clustering? Is there one?
- What are we looking for? How many clusters will we be well suited to our purpose?
- Which clustering algorithm can we choose?
- How can we label results of clustering? Are those clusters significant or even valid?
- Can we cluster the input data? Is there any trend for clustering in it?

From those questions arise a fundamental principle that all should take into account when designing a clustering process: each clustering algorithm imposes a view or a structure on input data and the main key to success when working with clustering algorithms will be to find a good match or combination between the clustering model and the input data ([96]).

If clustering is widely used in all scientific fields and for many different applications we also should mention that there are a gigantic amount of papers in the literature describing, refining, modifying or improving clustering techniques and it would be a real challenge to make an exhaustive list of those ones in a reasonable time. In the description of our custom clustering processing we will concentrate on describing one of the most common clustering techniques which we already talked about in previous sections, the *k-means* clustering algorithm.

b) Supervised learning

Once we went over the unsupervised learning part of the processing through feature selection and clustering, we also decided to design a machine learning tool based on a supervised learning approach.
Actually in most of automated diagnosis engines supervised learning is the key for providing a diagnosis proposition, it generally consists of the penultimate or last step of the processing plan of such decision making tool.

As mentioned in the overview of our work in the beginning of this document supervised learning can be considered as "the machine learning task of inferring a function from labelled training data" ([15]). This kind of learning algorithms is needed when we already know the features that will impact a specific decision type and when an instance of this decision type is needed. For example, a credit card company receives a large amount of applications for new cards and will be able, thanks to a supervised learning technique, to base its approval decision or classification on basis of previous cases of approval that went either well or wrong. In our context of automated diagnosis on basis of digital image analysis that would consist of working on the extracted and selected (or even clustered) features and to make a diagnosis decision emerge from a training phase on a large set of already diagnosed images (either normal or malignant).

As you could already understand, each supervised learning algorithm needs a training phase in which it can acquire knowledge of labelled input and construct a classification function. This function will be of course refined during this training period (which typically consists of iterations or stages). Once the function provides good enough results (which is measured by some test routine at the end of each iteration or stage with a threshold or a rule acting as the main decision system for this kind of checking) it can act as a powerful and automated classification model for further labelled instances (preferably out of the training dataset).

Once again there is a host of supervised learning techniques or algorithms and we couldn't make – if it was our purpose – an exhaustive list and description for each of them so as it was previously the case with clustering we won't be talking about SVMs, logistic regression, naive Bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees, boosted stumps, etc. (see the work in [94] and [95] for comprehensive listing and description of popular supervised learning techniques) . Instead we will focus on artificial neural network which is the supervised learning technique which remains the most widely used amongst similar studies and the one we have chosen to use in our project.

There are actually many types of artificial neural network: feed-forward ([97]), radial basis function network ([98]), Kohonen networks ([99]), learning vector quantization ([100]), recurrent neural networks ([101]), stochastic networks ([99]), etc. (See the systematic introduction on neural networks by Rojas in [99] for other types of network) In our work we will only focus on a specific type of neural network: the multilayer perceptron. Our explanations aim to give a solid understanding of this neural network in the perspective of its use in our work. They also summarize ideas and concepts coming from [99], for more details please refer to it.

At the very beginning of the design of such neural network is the McCulloch–Pitts model of logical gates that was already designed in the beginning of the twentieth century. From the rather basic computing unit inherited from Boolean models and electric circuits has derived a more complex type of computing unit which acts as the fundamental atom in a neural network. Figure 35 shows how we can represent this computing unit.

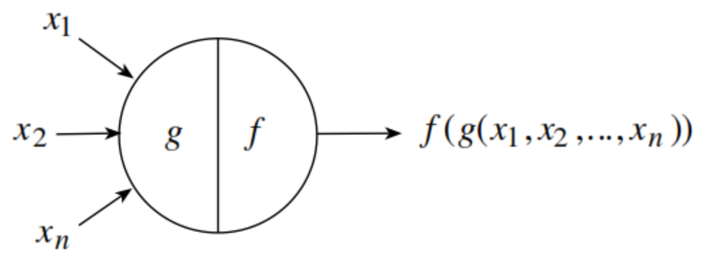


Figure 35 – Fundamental computing unit in a neural network ([99])

As represented in this figure, this computing unit takes several inputs x_i and produces an output $f(g(x_i))$ for all these inputs thanks to two functions f and g.
g is an integration function and computes a single values from the multiple inputs (it could be as simple as the sum function) whereas f is an activation function which takes the output from g as argument and computes a final output for the computing unit. We can also say that f acts as a threshold or firing function that will either produce result considering the inputs or discarding them in the overall computing of the network.
Also, in most neural networks the computing units are weighted, which means that the inputs will be assigned a weight before any computation at the unit-level.
Finally, we can also consider asynchronous network types where each computing units belonging to what we call a layer (a set of a computing units that are processed simultaneously in a neural network) has a state at a particular computing time in the network. This state information Q will then be used for the processing of the units in the next layer in the network. Figure 36 illustrates such computing unit.

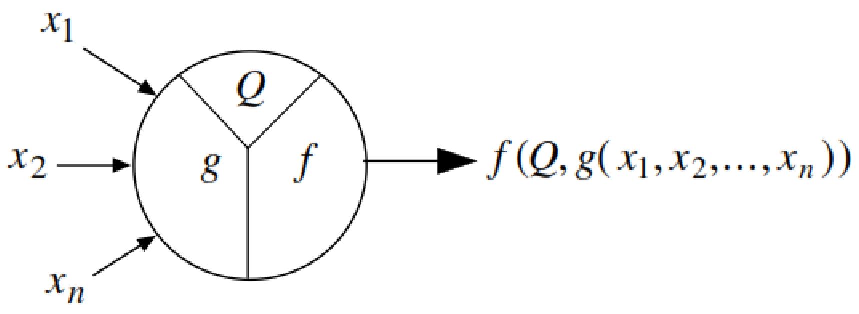


Figure 36 – Computing unit with state Q ([99])

A perceptron is actually the name given to the Minsky and Papert model back from 1969 which defines a neural network based on the introduced concept of computing units and layers, and a multi-layer perceptron is just a perceptron with multiple layers.

Now we have introduced the most basic concepts of a neural network, we have to introduce how it can actually learn to develop any decision efficiency.
A learning algorithm acts as a *self-adaptive method by which a network of computing units self-organizes to implement the desired behaviour*. Figure 37 shows the most basic structure of the learning process of a network.

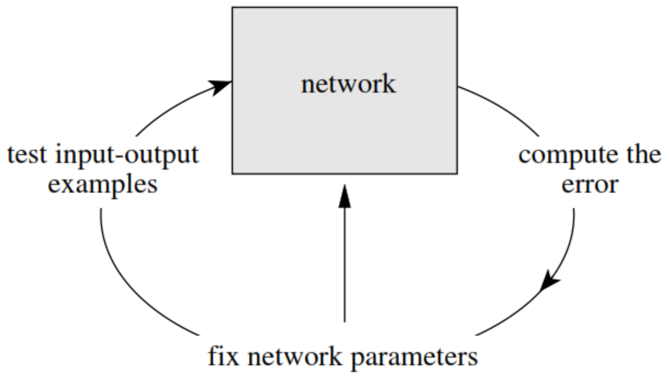


Figure 37 – Basic learning process of neural network ([99])

In some simple cases the weights for the computing units can be found through a sequential test of stochastically generated numerical combinations. However, such algorithms which look blindly for a solution do not qualify as "learning". A learning algorithm must adapt the network parameters according to previous experience until a solution is found, if it exists.

We also have to mention that a neural network can only be considered as supervised learning technique from the moment when the learning process of the network in indeed supervised, which means *that some input vectors are collected and presented to the network*. *The output computed by the network is observed and the deviation from the expected answer is measured. The weights are corrected according to the magnitude of the error in the way defined by the learning algorithm. This kind of learning is also called learning with a teacher, since a control process knows the correct answer for the set of selected input vectors*.

Supervised learning is further divided into methods which use reinforcement or error correction. Reinforcement learning is used when after each presentation of an input-output example we only know whether the network produces the desired result or not. The weights are updated based on this information (that is, the Boolean values true or false) so that only the input vector can be used for weight correction. In learning with error correction, the magnitude of the error, together with the input vector, determines the magnitude of the corrections to the weights, and in many cases we try to eliminate the error in a single correction step.

The perceptron learning algorithm is an example of supervised learning with reinforcement. Some of its variants use supervised learning with error correction (corrective learning).

c) Validation of results and diagnostic accuracy

Each scientific study has for requirement to use methods and techniques that will validate its results and build the trust for future consideration. As introduced in our overview that's obviously the case for our kind of work. Indeed a diagnostic engine has to prove its efficiency before emitting the possibility of its use in real situation.

There are once again a very large amount of available techniques to validate results in different kinds of situations. In our work the result validation consists mainly of model validation for the use of machine learning methods. We could also talk about the software testing process that is followed through the design and development of our software tools but we think of that as a common task that we therefore won't describe in this document. This choice is also explained by the fact that we often reused proven and well-tried libraries, software platforms and algorithms in the design and development phase for our tools. Therefore we don't get to explore this testing field even if it's of course a critical aspect in the development of such project.

In the field of medicine, it's common to evaluate the *diagnostic accuracy* of a diagnostic protocol, method or any other tool thanks to ROC curves ([102], [103], [104], [105], [106]). We therefore chose this methodology as a way to validate our results. This has been possible thanks to the already diagnosed images we received as a work basis from Alcides Chaux in our project (296 H&E stained prostate images – 98 normal, 198 malignant). In order to allow our readers to get a sufficient knowledge of this methodology we introduce some basics about diagnostic accuracy issues and the use of ROC (Receiver Operating Characteristic) curves.

From all times it has been quite difficult to even define what we mean by diagnostic accuracy. For sure one can have the intuition of a quality or efficiency derived feature assigned to a diagnosis but even with this intuition it can be hard to base a concrete analysis for a particular diagnostic protocol or method. Indeed several questions can arise when approaching diagnostic accuracy but they may all refer to the following ([105]): *how can we measure the quality of diagnostic information and of diagnostic decisions in a meaningful way?*

Answering to this question can actually serve to a twofold purpose: firstly, evaluating, improving or discarding a diagnostic protocol and, secondly, comparing diagnostic protocols between each other.

At the very core of the answer to this question are two concepts that are needed in order to outline a solution: *sensitivity* and *specificity* ([105]).

Sensitivity can be described as follows:

$$Sensitivity = \frac{\#\ true\ positive\ decisions\ (TP)}{\#\ positive\ cases}$$

And specificity can be described this way:

$$Specificity = \frac{\#\ true\ negative\ decisions\ (TN)}{\#\ negative\ cases}$$

Notice that in our case, *positive* would mean malignant state and *negative* would mean normal state of the studied image.

We can therefore say that sensitivity is concerned with finding all malignant cases and specificity is concerned with finding only malignant cases. By the way it is really important to note that sensitivity and specificity will be impacted by both the representation of data at the basis of the diagnosis and *threshold* that is considered to distinguish a positive case from a negative one in the distribution of cases.

The association of these two accuracy types can actually give a proper meaning to the diagnostic accuracy in general ([105]):

$$Accuracy = Sensitivity\ \times \frac{\#\ positive\ cases}{all\ cases} + Specificity\ \times \frac{\#\ negative\ cases}{all\ cases}$$

As you may have probably already noticed, sensitivity is just what we commonly call the TP rate (or recall) and specificity the TN rate. In the same manner we can also define the FP (False Positive) rate and the FN (False Negative) rate as other key indicator for accuracy.

If we now want to make the decision threshold (the same as we have just talked about earlier) to see how those values evolve, we basically build what is called a ROC curve. For example, if we plot the TP and FP rates as the x and y axis units of a Cartesian frame, we can obtain the ROC curve by just inserting the TP and FP pairs generated at each threshold decision variation.

Thus, from a conventional two-by-two table with TP, TN, FP and FN it is possible to both evaluate the accuracy of diagnostic method or protocol and to compare it to others. *The most desirable property of ROC analysis is that the accuracy indices derived from this technique are not distorted by fluctuations caused by the use of arbitrarily chosen decision criteria or cut-offs*. *[…] The derived summary measure of accuracy, such as the area under the curve (AUC) determines the inherent ability of the test to discriminate between the diseased and healthy populations (21). Using this as a measure of a diagnostic performance, one can compare individual tests or judge whether the various combination of tests (e.g. combination of imaging techniques or combination of readers) can improve diagnostic accuracy.* ([106]).

In order to allow the less informed reader to grasp knowledge from those explanations, Figure 1 shows two distribution functions, one for the normal cases and the other one for the malignant cases as well as different decision thresholds.

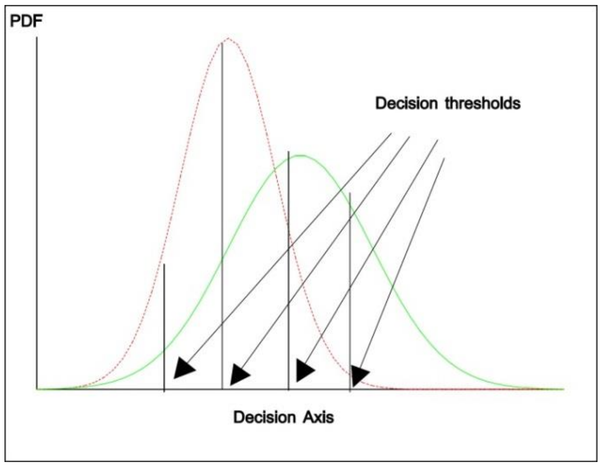


Figure 38 – Two overlapping cases distributions with four different decision thresholds ([106])

Figure 2 now shows a resulting ROC curves from those observations.

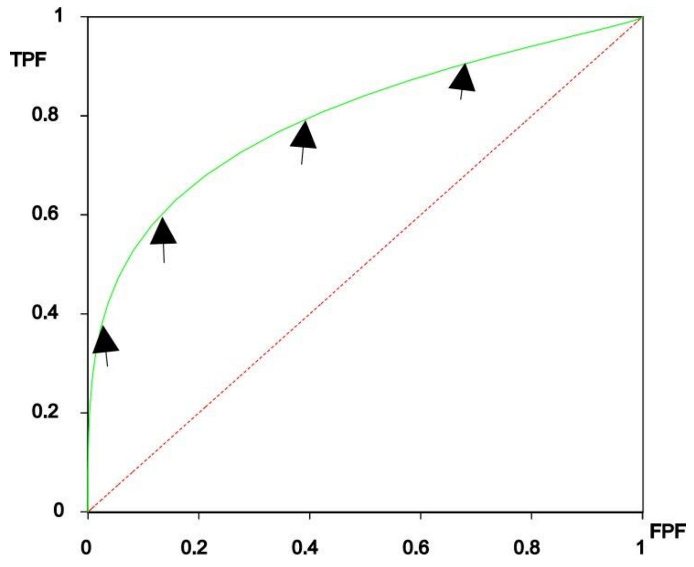


Figure 39 – ROC curve resulting from the overlapping distributions in Figure 38 ([106])

Finally we have to say a few words about the interpretation that can arise from the reading of ROC curves. For this interpretation call, we refer to the work in [106]:

The plot of TPF (sensitivity) versus FPF (1-specificity) across varying cut-offs generates a curve in the unit square called an ROC curve. ROC curve corresponding to progressively greater discriminant capacity of diagnostic tests are located progressively closer to the upper left-hand corner in "ROC space" (figure 39 shows test B has a greater discriminate capacity than test A). An ROC curve lying on the diagonal line reflects the performance of a diagnostic test that is no better than chance level, i.e. a test which yields the positive or negative results unrelated to the true disease status. The slope of an ROC curve at any point is equal to the ratio of the two density functions describing, respectively, the distribution of the separator variable in the diseased and non-diseased populations, i.e. the likelihood ratio. A monotonically increasing likelihood ratio corresponds to a concave ROC curve. The area under the curve (AUC) summarizes the entire location of the ROC curve rather than depending on a specific operating point. The AUC is an effective and combined measure of sensitivity and specificity that describes the inherent validity of diagnostic tests.

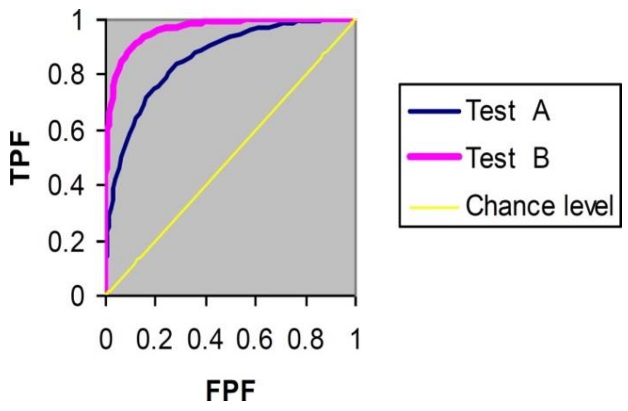


Figure 40 – ROC curves for tests A and B as well as the Chance level ([106])

Despite this overall consensus about the AUC feature for the analysis of ROC curves it should be noted that in case of overlapping or crossing curves (corresponding to different tests) this feature is not the best suited to provide satisfactory results. However, according to any particular purpose when analysing such curves, it is also possible to refer to the TP-FP pairs to base the final decision for the choice or the approval of a diagnostic method or protocol.

Finally, ROC curves are not the ultimate solution for diagnostic accuracy computation. Even if it is widely used and recognized as a very efficient tool, it suffers from some not-always-overcome problems ([107]):

- *Lack of gold standard for diagnosis*;
- *Lack of reproducibility* (disagreement among pathologists for example);
- *Bias in sample selection, spectrum of disease used in evaluating test* (are the worst cases analysed? Are the cases more *standard*? Is the case distribution balanced between normal and malignant cases?)
- *Can't always reliably measure ROC area* (see our warning just above).

B. Custom machine learning processes

a) Clustering process

Once we extracted and selected the features (with the PCA approach) at the last stage of processing, we decided to perform some clustering on the dataset we obtained as a result. This decision was motivated by one potential and particular advantage that a clustering process could have for the learning stage of a diagnostic engine suite: understanding the underlying structure within our computed set thanks to the “natural” groups distinguished by the clustering algorithm. More specifically, how can we learn from what we have extracted and selected in the H&E stained images we had processed?

As announced in the state of the art section above we decided to use one of the most common clustering techniques, which is k-means clustering. We briefly introduce the basic principles of this technique and rely on the work in [74] and [108] for this purpose.

The idea of behind the k-means algorithm is to consider a number of clusters with a random seed for each of them. Then the algorithm will iteratively computes the distance (Euclidean, Manhattan, Chebyshev, etc.) between those seeds and each item in the data collection given in input. From the second iteration to the last one, seeds evolves in centroids, computed from the mean value of features for items in the cluster. The algorithm stops once an acceptance threshold is reached; this threshold is generally computed as the number of cluster assignation changes on the total number of item in the collection at the end of every iteration.

Here's a more formal version of the algorithm, taken from [74]:

```
1. Choose k initial cluster centres (for example, k random points)
2. Repeat (until centres don't change)
        - Assign instances to clusters based on distance to cluster centres
        - Recompute centres by computing the centroids of clusters
```

As you can see now the algorithm is really simple and can be implemented without reaching a high complexity and these two factors surely contribute to the large success of this particular clustering technique, which is not the case for its accuracy (it can sometimes be really bad). Also, it has to be noted that the k-means algorithm can be improved both on the performance and accuracy sides by selecting appropriate seeds when it is feasible (k-means++ method described in [108]).

For most of our machine learning processes we relied on the Weka platform (already introduced in the feature selection section) and those processes include the clustering one. We have tried several configurations of the k-means algorithm but we finally stick to its most basic version, using the Euclidean distance as the distance metric and a random seed initialization phase which has proven to give the best results (even compared to the k-means++ seed initialisation, which was quite a surprise for us). Of course, as we wanted to observe if there was a natural distinction between malignant and normal images, we first set the number of clusters to two. Finally, and this will be the case for all our machine learning tests, we decided to balance our initial dataset of images with 98 normal images and 98 malignant images (we had actually 98 normal images and 198 malignant ones but we couldn't get more normal images so we had to perform this selection for our tests).

Table 5 shows the results obtained after the clustering of the pipe-1-processed dataset (without PCA).

Table 5 – Clustering results for pipe-1-processed dataset without PCA

# Clusters	2
Clustered instances	96 instances in 1st cluster (Normal) 100 instances in 2nd cluster (Malignant)
Cluster distribution	1st cluster: 47 malignant & 49 normal 2nd cluster: 51 malignant & 49 normal
Incorrectly clustered instances	1st cluster: 47 → 48.9583% 2nd cluster: 49 → 49% Total:96 → 48.9796%

Table 6 shows the results obtained after the clustering of the pipe-1-processed dataset (with PCA).

Table 6 - Clustering results for pipe-1-processed dataset with PCA

# Clusters	2
Clustered instances	100 instances in 1st cluster (Normal) 96 instances in 2nd cluster (Malignant)
Cluster distribution	1st cluster: 51 malignant & 49 normal 2nd cluster: 68 malignant & 40 normal
Incorrectly clustered instances	1st cluster: 30 → 34.0909% 2nd cluster: 40 → 37.037% Total:70 → 35.7143%

Table 7 shows the results obtained after the clustering of the pipe-2-processed dataset (without PCA).

Table 7 - Clustering results for pipe-2-processed dataset without PCA

# Clusters	2
Clustered instances	82 instances in 1st cluster (Normal) 114 instances in 2nd cluster (Malignant)
Cluster distribution	1st cluster: 31 malignant & 51 normal 2nd cluster: 51 malignant & 47 normal
Incorrectly clustered instances	1st cluster: 31 → 37.8049% 2nd cluster: 47 → 41.2281% Total:78 → 39.7959%

Table 8 shows the results obtained after the clustering of the pipe-2-processed dataset (with PCA).

Table 8 - Clustering results for pipe-2-processed dataset with PCA

# Clusters	2
Clustered instances	66 instances in 1st cluster (Normal) 130 instances in 2nd cluster (Malignant)
Cluster distribution	1st cluster: 20 malignant & 46 normal 2nd cluster: 78 malignant & 52 normal
Incorrectly clustered instances	1st cluster: 20 → 30.3030% 2nd cluster: 52 → 40% Total:72 → 36.7347%

Table 9 shows the results obtained after the clustering of the pipe-3-processed dataset (without PCA).

Table 9 - Clustering results for pipe-3-processed dataset without PCA

# Clusters	2
Clustered instances	105 instances in 1st cluster (Normal) 91 instances in 2nd cluster (Malignant)
Cluster distribution	1st cluster: 38 malignant & 67 normal 2nd cluster: 60 malignant & 31 normal
Incorrectly clustered instances	1st cluster: 38 → 36.1905% 2nd cluster: 31 → 34.0659% Total:69 → 35.2041%

Table 10 shows the results obtained after the clustering of the pipe-3-processed dataset (with PCA).

Table 10 - Clustering results for pipe-3-processed dataset with PCA

# Clusters	2
Clustered instances	85 instances in 1st cluster (Normal) 111 instances in 2nd cluster (Malignant)
Cluster distribution	1st cluster: 20 malignant & 46 normal 2nd cluster: 78 malignant & 52 normal
Incorrectly clustered instances	1st cluster: 20 → 30.3030% 2nd cluster: 52 → 40% Total:72 → 36.7347%

As it was expected we can see how the feature selection process can really improve this clustering process. However, and quite surprisingly, that was not the case for the pipe-3-processed dataset which proves to give better clustering results without the feature selection process.

Also, it has to be noticed that we obtained poor results in terms of classification in both cases with a maximal accuracy of about 65%. We assume here that those results can be imputed to the simplicity of the k-means algorithm which may not be the most well suited algorithm for highly multidimensional data and the data representation that don't really allow to distinguish 2 really distinct clusters.

Finally, we can observe that 5 out of our 6 clustering experiments had better chance to classify normal images than malignant images.

Without a doubt, this clustering process should be improved in future work. Some tracks of improvement could be the refinement of the selected features, the choice for a more suited clustering algorithm and an analysis whose purpose would be to determine the most appropriate data representation for the clustering process to give more satisfactory results.

b) Supervised learning process

As mentioned in the state of the art section for machine learning we decided to opt for a multilayer perceptron model as the main supervised learning technique in the design of our diagnostic engine tool suite.

Once again we relied on the Weka platform to build and use the model as a classifier on our 3 main datasets. For each of these datasets we had to tune the characteristics of the basis perceptron model (learning rate, momentum rate, etc.).

In this section we provide the details and results of our experiments with the multilayer perceptron classifier applied to our 3 datasets. It has to be mentioned that we only worked with datasets that have been previously processed by the PCA feature selection algorithm.

Table 11 shows the details for the classification of the pipe-1-processed dataset with a multilayer perceptron (percentage split of 70% of the 196 images, that means that 30% of the images will be used as test data).

Vocabulary:

- Learning rate: rate to which the weights are updated.
- Momentum: momentum applied to the weights during updating.
- Hidden layers: computed as ((#features + #classes)/2)

Table 11 – Details for the classification of the pipe-1-processed dataset with a multilayer perceptron

Learning rate	0.15
Momentum	0.3
Hidden layers	17
Training iterations	500
Percentage split	70%

Table 12 shows the results for the classification of the pipe-1-processed dataset with a multilayer perceptron.

Table 12 - Results for the classification of the pipe-1-processed dataset with a multilayer perceptron (70% percentage split)

TP Rate	FP Rate	Precision	Recall	Class
0.821	0.100	0.941	0.821	Malignant
0.900	0.179	0.720	0.900	Normal
0.847	0.127	0.866	0.847	Average

Figure 41 shows the ROC curve for the classification of the pipe-1-processed dataset with a multilayer perceptron.

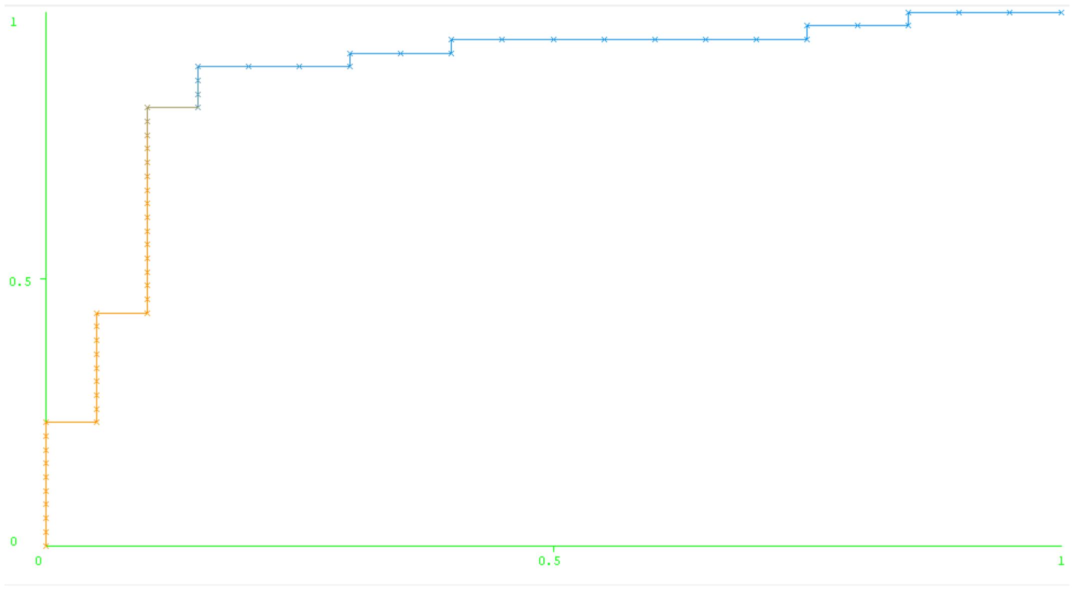


Figure 41 – ROC curve for classification of the pipe-1-processed dataset with a multilayer perceptron (malignant class)

When we tried to increase the percentage split to 90% (test set consisting of 20 images), we reached 18 out of 20 correctly classified instances and with a percentage split of 95% (test set consisting of 10 images), all the instances were classified correctly. Table 13 shows the results for 90% percentage split.

Table 13 - Results for the classification of the pipe-1-processed dataset with a multilayer perceptron (90% percentage split)

TP Rate	FP Rate	Precision	Recall	Class
0.846	0	1	0.846	Malignant
1	0.154	0.778	1	Normal
0.9	0.054	0.922	0.9	Average

We can once again observe that normal cases are more easily classified with this type of classifier.

Table 14 shows the details for the classification of the pipe-2-processed dataset with a multilayer perceptron.

Table 14 - Details for the classification of the pipe-2-processed dataset with a multilayer perceptron

Learning rate	0.35
Momentum	0.3
Hidden layers	17
Training iterations	500
Percentage split	76%

Table 15 shows the results for the classification of the pipe-2-processed dataset with a multilayer perceptron.

Table 15 - Results for the classification of the pipe-2-processed dataset with a multilayer perceptron (76% percentage split)

TP Rate	FP Rate	Precision	Recall	Class
0.781	0.133	0.926	0.781	Malignant
0.867	0.219	0.65	0.867	Normal
0.809	0.161	0.838	0.809	Average

Figure 42 shows the ROC curve for the classification of the pipe-2-processed dataset with a multilayer perceptron.

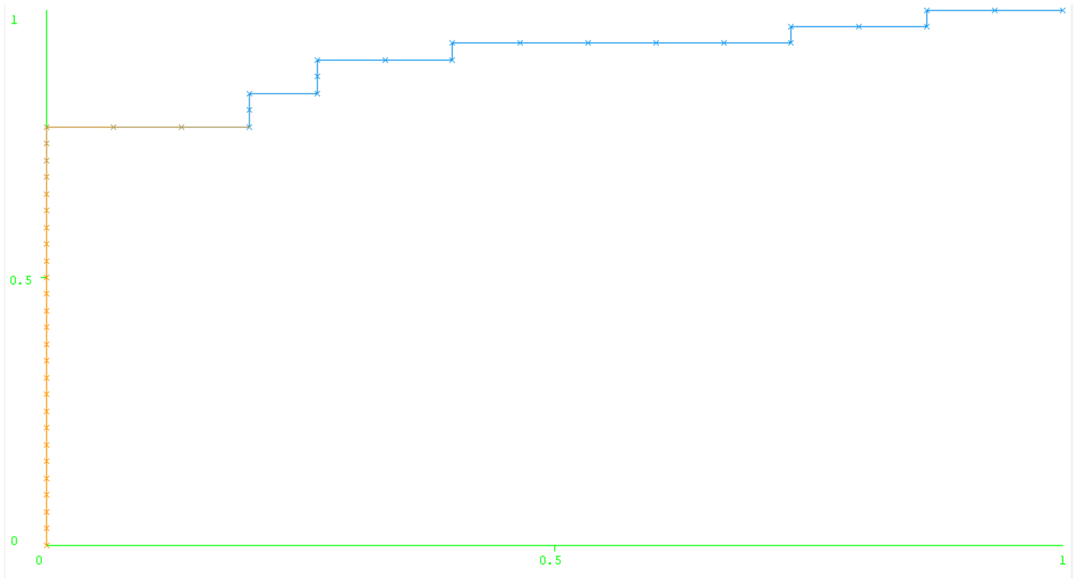


Figure 42 - ROC curve for classification of the pipe-2-processed dataset with a multilayer perceptron (malignant class)

Once again, if we increase the percentage split to 90%, we reached 85% of correctly classified instances and with 95%, we reached 90% of correctly classified instances. We can thus see here that the second dataset is not as prone as the first one for classification performance but the results are still acceptable at this stage of research.

Table 16 shows the details for the classification of the pipe-3-processed dataset with a multilayer perceptron.

Table 16 - Details for the classification of the pipe-3-processed dataset with a multilayer perceptron

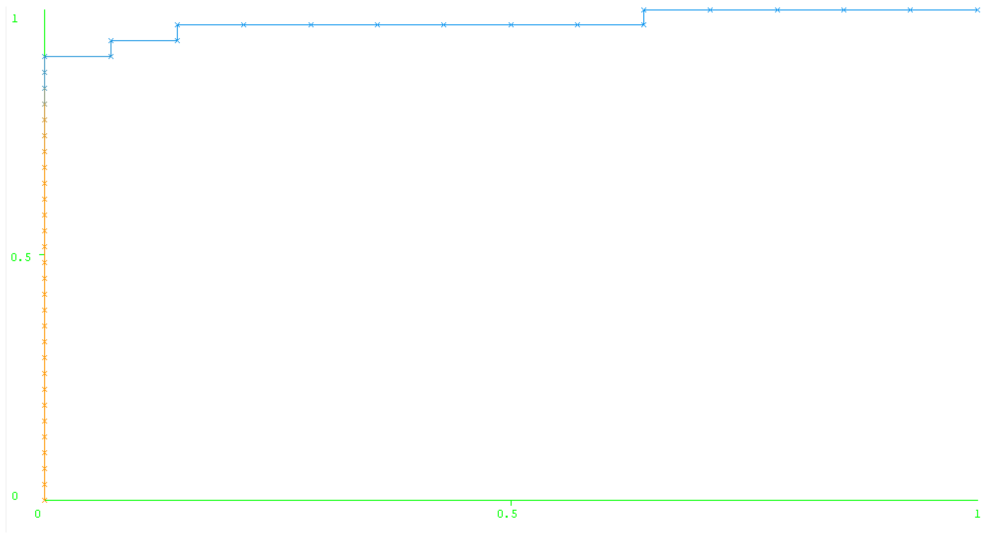
Learning rate	0.35
Momentum	0.3
Hidden layers	17
Training iterations	500
Validation Threshold	30
Percentage split	77%

Table 17 shows the results for the classification of the pipe-3-processed dataset with a multilayer perceptron.

Table 17 - Results for the classification of the pipe-3-processed dataset with a multilayer perceptron (77% percentage split)

TP Rate	FP Rate	Precision	Recall	Class
0.774	0	1	0.774	Malignant
1	0.226	0.667	1	Normal
0.844	0.07	0.896	0.844	Average

Figure 43 shows the ROC curve for the classification of the pipe-3-processed dataset with a multilayer perceptron.



Again, when tuned with a percentage split of 90%, the classifier returns 90% of correctly classified instances and with a percentage split of 95%, we reached the same results.

c) Discussion about datasets and supervised learning

As it can been seen from the previous results it's quite difficult to come with a clear winner as for the combination of a dataset with its multilayer perceptron classifier. If we stick to what is commonly used to compare diagnostic protocols we could refer to the ROC curves and use the ROC areas to order the three experiments by their evaluated accuracy. If we would actually do that we would have, for the same percentage split of 70% (which is a reasonable and quite common value for this kind of tests), the third experiment first with a ROC area of 0.944, then the first experiment with a ROC area of 0.881 and, finally, the second experiment with a ROC area of 0.853.

But at this point we fell the urge to nuance those results as it has been previously the case throughout the project. Because indeed if we can pinpoint the best ROC area we can also see that the first pipeline is the only one reaching a perfect classification with a high percentage split whereas the two others seem to reach a performance threshold after 90% of percentage split.

Also, when we performed cross-validation for our three experiments we quickly observed that none of our dataset-classifier combinations (with multilayer perceptron) could reach more than 85% of correctly classified instances over our 196 images dataset.

Finally, at this stage of the project, we are quite satisfied with those results. Of course there is a large improvement margin but as it is a first attempt at designing image processing and machine learning tools in order to provide an open-source basis for collaborative research on cancer diagnosis, we couldn't feel anything but motivation to continue this work and further refine it. The tracks for improvement regarding the supervised learning phase are numerous and can be tracked down from the refinement of the image processing tools to the selection of more suited supervised learning technique (at this stage of the project, we already tried the *logistic regression model* but without getting any better results) not forgetting the adaptation of data representation to clustering and supervised learning techniques or using the clustering process to divide the whole dataset into refined learning sets.

6. Conclusion and Future Work

We would like to finish the first step of this research project here by reminding what the main contributions were and pinpointing some further research directions for the following steps.

First, we provide a large amount of references, pointers and explanations about state-of-the-art techniques, methods and models related to image processing and machine learning in the service or automated diagnostic engines. These elements consist in the first attempt at making a transdisciplinary knowledge reference for future development of a new collaborative research project.

Then, we provide some software tools in what we can consider as the embryo for an open-source diagnostic engine tool suite that is aimed to be published, edited, improved and/or refined online once the project and those tools will have matured. Despite our lack of experience with particular fields in the development of the provided tools we could provide tools with a respectable degree of genericity, accuracy and even sometimes, conceptual complexity.

These two major contributions of our work relied on more than 100 literature references and on several free and open software platforms which were critical to this project (namely ImageJ for image processing and Weka for machine learning).

Of course, several tracks for development and improvement have been given in this document and we won't remind them in details but we will instead give the general directions that characterize them.

As image processing is concerned future work could consist to optimize the developed pipelines thanks to local or more general optimizations. Also it could be interesting to build a pattern recognition process that would refer to already detected cells to find new ones.

As for machine learning, we think that data representation, the extraction of other multiple features and the selection of a wide range of unsupervised and supervised learning tools could contribute to a wider perspective on this field of knowledge and achieve better results than those we have currently reached. Moreover some benchmarks for cell characterization in normal and malignant images are still to be established.

Also, this work wouldn’t be what it is meant to be if an online platform wouldn’t be built, referencing our work, our datasets and our tools as well as leaving them open for comments, requests and further improvement by a worldwide community of specialists.

Finally, after this almost one-year project we can have a better perspective on the work that we have made and how we would make it today from scratch and actually, it isn’t far from what we did. Of course, we would be more rigorous on particular points (maybe provide an even more exhaustive list of references) and take more time for the machine learning phase, which only consisted in one month for the internship and two months of other sparse research work but overall we think that the methodology we adopted for this work is quite the most common and effective one, starting with what has already been done and evolving towards something custom or even new, would that be methodological or technical.

Bibliography

[1] Globocan 2012 : Estimated Cancer Incidence, Mortality and Prevalence Worldwide in 2012. http://globocan.iarc.fr/Pages/fact_sheets_cancer.aspx. [Online; accessed 08-February-2014].

[2] L. Krnjacki, P. Baade, D. Youlden. "International epidemiology of prostate cancer: Geographical distribution and secular trends". In: Molecular Nutrition and Food Research, 53.2 (2009), pp. 171–184.

[3] O. Brawley, R. Siegel, E. Ward and A. Jemal. "Cancer statistics, 2011". In: CA: A Cancer Journal for Clinicians, 61.4 (2011), pp. 212–236.

[4] Aging and Cancer. http://www.cancer.net/navigating-cancer-care/olderadults/aging-and-cancer. [Online; accessed 09-February-2014].

[5] J. Ma, M. Teverovskiy, V. Kumar and A. Kotsianti. "Improved Prediction of Prostate Cancer Recurrence Based on an Automated Tissue Image Analysis System". In: IEEE International Symposium on biomedical imaging: Nano to Macro, 1 (2004), pp. 257–260.

[6] H. Pang, A. Tabesh, V. Kumar and D. Verbel. "Automated Prostate Cancer Diagnosis and Gleason Grading of Tissue Microarrays". In: Medical Imaging 2005: Image Processing, 5747 (2005), pp. 58–70.

[7] S. Osowski, M. Kruk and R. Koktysz. "Numerical characterization of the images of prostate cancer for recognition of Gleason scale". In: Przeglad Elektrotechniczny, 5 (2011).

[8] C. Demir and B. Yener. "Automated cancer diagnosis based on histopathological images: a systematic survey". In: Technical Report, Rensselaer Polytechnic Institute, Department of Computer Science, (2009), pp. 12–14.

[9] S.J. Kim, A.W. Wetzel, R. Crowley and R. Dawson. “Evaluation of prostate tumor grades by content based image retrieval”. In: 27th AIPR Workshop: Advances in Computer Assisted Recognition, 244 (1999).

[10] W. Christens-Barry and A. Partin. “Quantitative Grading of Tissue and Nuclei in Prostate Cancer for Prognosis Prediction”. In: Johns Hopkins APL Technical Digest, 18.2 (1997), pp. 226–233.

[11] M. Gao and P. Bridgman. “Computer Aided Prostate Cancer Diagnosis Using Image Enhancement and JPEG2000”. In: Proceedings of SPIE Annual Meeting, 5203 (2003), pp. 323–334.

[12] I. Guyon and A. Elisseeff. “An Introduction to Variable and Feature Selection”. In: Journal of Machine Learning Research, 3 (2003), pp. 1157–1182.

[13] T. Sejnowski, G. Hinton. “Unsupervised Learning: Foundations of Neural Computation”. 1st edition. MIT Press, (1999).

[14] P. Dayan. “Unsupervised Learning”. In: The MIT Encyclopedia of the Cognitive Sciences, (1999).

[15] A. Rostamizadeh, M. Mohri and A. Talwalkar. “Foundations of Machine Learning”. 1st edition. MIT Press, (2012).

[16] The Science and Application of Hematoxylin and Eosin Staining. http://www.feinberg.
northwestern.edu/research/docs/cores/mhpl/HandE_troubleshooting.pdf.
[Online; accessed 11-February-2014].

[17] P.W. Swindle, D. Scherr and P.T. Scardino. “National Comprehensive Cancer Network Guidelines for the Management of Prostate Cancer”. In: Urology, 61.2 (2003), pp. 14–24.

[18] F. Darro et al. “Characterization of the Differentiation of Human Colorectal Cancer Cell Lines by Means of Voronoï Diagrams”. In: Cytometry, 14 (1993), pp. 783–792.

[20] R. Marcelpoil J. Sudbo and A. Reith. “New algorithms based on the Voronoï Diagram applied in a pilot study on normal mucosa and carcinomas”, (2000).

[21] B. Yener, C. Demir and S. Gultekin. “The cell graphs of cancer”. In: Bioinformatics, 20. 1 (2004), pp. 145–151.

[22] S. Gultekin, C. Demir and B. Yener. “Spectral analysis of cell graphs of cancer”, (2005).

[23] J. Zahm et al. “Quantitative videomicoscopic analysis of the sociologic behavior of non-invasive and invasive tumor cell lines”. In: Cellular and Molecular Biology, 52.6 (2007), pp. 54–60.

[24] C. Nagic, C. Bilgin, C. Demir and B. Yener. “Cell-Graph Mining for Breast Tissue Modelling and Classification”, (2007).

[25] J. Konsti. “Automated Image Analysis of Cancer Tissue Adapted for Virtual Microscopy”, (2012).

[26] Y. Bin, S.G. Chang and M. Vetterli. “Adaptive wavelet thresholding for image denoising and compression”. In: IEEE Transactions on Image Processing, 9.9 (2002), pp. 1532– 1546.

[27] M. Biswas and H. Om. “A New Soft-Thresholding Image Denoising Method”. In: Procedia Technology, 6 (2012), pp. 10–15.

[28] E. Meijering. “Cell Segmentation: 50 Years Down the Road”. In: IEEE Signal Processing Magazine, 29.5 (2012), pp. 140–145.

[29] F. Merchant, Q. Wu and K. Castleman. “Microscope Image Processing”, 1st edition, Academic Press, (2008).

[30] P. Soille, L. Vincent. “Watersheds in digital spaces: An efficient algorithm based on immersion simulations”. In: IEEE T. Pattern Anal., 13 (1991), pp. 583–598.

[31] A. Samborskiy. “Cell Detection and Counting for Microscope Images with Applications to Stem Cell Engineering”. In: Carnegie Mellon University - Department of Electrical and Computer Engineering.

[32] A. Kaspers. “Blob Detection”. In: Biomedical Image Sciences - Image Sciences Institute.

[33] M. Gupta. “Cell Identification by Blob Detection”. In: Proc. of the Intl. Conf. on Advances in Computer Science and Electronics Engineering, (2012).

[34] J. Byun et al. “Automated tool for nuclei detection in digital microscopic images: Application to retinal images”. In: Molecular Vision 12 (2006), p. 949.

[35] T. Lindeberg. “Detecting salient blob-like image structures and their scales with a scalespace primal sketch: a method for focus-of-attention”. In: International Journal of Computer Vision 11.3 (1993), pp. 283–318.

[36] R. Boyle, M. Sonka, V. Hlavac. “Processing, Analysing, and Machine Vision. Thomson”, (2008).

[37] S. Beucher and C. Lantuéjoul. “Use of watersheds in contour detection”. In: International workshop on image processing, real-time edge and motion detection, (1979).

[38] L. G. Minor and J. Sklansky. “Detection and segmentation of blobs in infrared images”. In: IEEE Transactions on Systems, Man and Cybernetics , 11.3 (1981).

[39] T. Lindeberg. “Feature detection with automatic scale selection”. In: International Journal of Computer Vision, 30.2 (1998), pp. 79–116.

[40] S. Hinz. “Fast and Subpixel Precise Blob Detection and Attribution”. In: Proceedings ICIP 2005, 3 (2005), pp. 457–460.

[41] C. Damerval and S. Meignen. “Blob detection with wavelet maxima lines”. In: IEEE Signal Processing Letters, 14.1 (2007), pp. 39–42.

[42] P.-E. Forssün and G. Granlund. “Robust multiscale extraction of blob features”. In: Proceedings of the 13th Scandinavian Conference on Image Analysis, 2749 (2003), pp. 11–18.

[43] H.E. Danielsen, B. Nielsen, F. Albregtsen. “The use of fractal features from the periphery of cell nuclei as a classification tool”. In: Anal. Cell. Pathol., 19 (1999), pp. 21–37.

[44] A.J. Einstein et al. “Reproducibility and accuracy of interactive segmentation procedures for image analysis in cytology”. In: J. Microsc. ,188 (1997), pp. 136–148.

[45] M. Kass, A. Witkin, and D. Terzopoulos. “Snakes: Active Contour Models”. In: International Journal of Computer Vision, (1987), pp. 321– 331.

[46] A. Sheehy et al. “Region and contour based cell cluster segmentation algorithm for in situ microscopy”. In: Electrical Engineering, Computing Science and Automatic Control, 2008. CCE 2008. 5th International Conference on (2008), pp. 168–172.

[47] J. Iguelmar Miranda. “Anisotropic Diffusion Model for Edge Detection: a Java Implementation”, 1st edition, (2008).

[48] R. Collins. “Introduction to Computer Vision - Lecture 04: Smoothing”. http://www.cse.psu.edu/~rcollins/CSE486/lecture04.pdf. [Online, accessed 11-February-2014].

[49] P. Perona and J.Malik. “Scale-Space and Edge Detection Using Anisotropic Diffusion”. In: IEEE Transactions On Pattern Analysis and Machine Intelligence, 12.7 (1990), pp. 629–639.

[50] A. Majumder and S. Irani. “Perception-Based Contrast Enhancement of Images”. In: CM Trans. Appl. Percpt., 4.3 (2007).

[51] ImageJ Official Documentation - Contrast Enhancement Process. [Online; accessed 11February-2014].

[52] A. Ruifrok and D. Johnston “Quantification of Histochemical staining by Colour Deconvolution”. In: Analytical and Quantitative Cytology and Histology Journal, 23.4 (2001), pp. 291–299.

[53] Lode’s Computer Graphics Tutorial - Flood Fill. http://lodev.org/cgtutor/ floodfill.html. [Online; accessed 08-February-2014].

[54] Explanation of the LAB Colour Space. http://www.aces.edu/dept/fisheries/ education/pond_to_plate/documents/ExplanationoftheLABColourSpace.pdf. [Online; accessed 08-February-2014].

[55] CIELab colour space. http://colourbrate.co.za/wp/wp-content/uploads/2013/07/ kompozice7_space-cieLAB.jpg. [Online; accessed 08-February-2014].

[56] Delta E colour distance. http://w3.efi.com/services/proofing-services/\~/ media/Files/EFI/COM/Services/Delta\%20E_H_T.pdf. [Online; accessed 08February-2014].

[57] R.M. Haralick. “Statistical and structural approaches to texture”. In: Proc. of IEEE. , 67 (1979), pp. 786–804.

[58] M.M. Galloway. “Texture analysis using gray level run lengths”. In: Computer Graphics and Image Processing, 4 (1975), pp. 172–179.

[59] R.M. Haralick, K. Shanmugam, and Its’Hak Dinstein. “Textural Features for Image Classification”. In: IEEE Transactions on Systems, Man and Cybernetics, 3.6 (1973), pp. 610–621.

[60] S.G. Mallat. “A theory for multiresolution signal decomposition: the wavelet representation”. In: IEEE Transactions on Pattern Analysis and Machine Intelligence , 11.7 (2002), pp. 674–693.

[61] Benoit B. Mandelbrot. “The Fractal Geometry of Nature”, 1st edition, W. H. Freeman and Company, (1982).

[62] Cross SS. “Fractals in pathology.” In: J. Pathol, 1 (1997), pp. 1–8.

[63] Heymans O et al. “Is fractal geometry useful in medicine and biomedical sciences?” In: J. Pathol, 54.3 (2000), pp. 360–366.

[64] S.K. Mohanty, P. Dey “Fractal dimensions of breast lesions on cytology smears”. In: Diagn Cytopathol, 29.2 (2003), pp. 85–86.

[65] A.N. Esgiar et al. “Fractal analysis in the detection of colonic cancer images”. In: IEEE Transactions on Information Technology in Biomedicine, 6.1 (2002), pp. 54–58.

[66] A. S. Kerenji et al. “Fractal dimension of hepatocytes’ nuclei in normal liver vs hepatocellular carcinoma (HCC) in human subjects - preliminary results”. In: Archive of Oncology 8.2 (2000), pp. 47–50.

[67] P. Dey, L. Rajesh. “Fractal dimensions in urine smears: a comparison between benign and malignant cells”. In: Anal Quant Cytol Histol., 25.3 (2003), pp. 181–182.

[68] R. Sedivy et al. “Fractal analysis: an objective method for identifying atypical nuclei in dysplastic lesions of the cervix uteri”. In: Gynecol Oncol., 75.1 (1999), pp. 78–83.

[69] A.J. Einstein, H.S. Wu, J. Gil and M. Sanchez. “Fractal characterization of chromatin appearance for diagnosis in breast cytology”. In: J Pathol., 185.4 (1998), pp. 366–381.

[70] B. Nawrocky Raby et al. “Quantitative Cell Dispersion Analysis: New Test to Measure Tumor cell Aggressiveness”. In: Int. J. Cancer, 93 (2001), pp. 644–652.

[71] Y. Bin, S.G. Chang and M. Vetterli. “Noise Reduction for Magnetic Resonance Images via Adaptive Multiscale Products Thresholding”. In: IEEE Transactions on Medical Imaging, 22.9 (2003), pp. 1532–1546.

[72] A. Chydzinskia, B. Smolka, K.N. Plataniotisb et al. “Self-adaptive algorithm of impulsive noise reduction in colour images”. In: Pattern Recognition, 35 (2002), pp. 1771–1784.

[73] Colour Difference. http://en.wikipedia.org/wiki/Colour_difference. [Online; accessed 22-October-2013].

[74] Ian H. Wittten & Eibe Frank. “Data Mining - Practical Machine Learning Tools and Techniques. 2nd edition” In : Weka, (2005).

[75] R.Marcepoil, Y.Usson, J.M.Chassery. “Segmentation Morphologique incluant des paramètres d’ordre et désordre : Quantification par Diagramme de Voronoï et application à la sociologie cellulaire”. In: AFCET RFIA, 8e Congrès (1991), pp. 967 – 972.

[76] N. Merzougui. “Un algorithme évolutionnaire pour la segmentation d'images basé sur le diagramme de Voronoï” In : Université Kasdi Merbah-Ouargla, (2012).

[77] S. Wold, K. Esbensen, P. Geladi. “Principal Component Analysis”. In: Chemometrics and Intelligent Laboratory Systems, 2 (1987), pp. 37 – 52.

[78] I. Guyon, A. Elisseeff. “An Introduction to Variable and Feature Selection”. In: Journal of Machine Learning Research, 3 (2003), pp. 1157 – 1182.

[79] Kan Deng. “Omega: on-line memory-based general purpose system classifier – Chapter 7: Feature Selection” (1998), pp. 117 – 119.

[80] A. Kalousis, J. Prados, P. Nguyen, M. Hilario. “Stability of Feature Selection Algorithms”, (2009), pp. 1 – 7.

[81] S. Fortune. “A Sweepline Algorithm for Voronoï Diagrams”. In: Proceedings of the second annual symposium on computational geometry, (1986), pp. 313 – 322.

[82] L. Guibas, J. Stolfi. “Primitives for the manipulation of general subdivisions and the computation of Voronoï”. In: ACM Transactions on Graphics (TOG) 14.2 (1992), pp. 74 – 123.

[83] L. He, L. Rodney Long, S. Antani, and G. Thoma. “Histology image analysis for carcinoma detection and grading”. In: Computational Methods Programs Biomed. 107.3 (2012), pp. 538 – 556.

[84] Agner S, Madabhushi A, Rosen M, Schnall M, Nosher J, Somans S, et al. “A comprehensive multi-attribute manifold learning scheme-based computer aided diagnostic system for breast MRI.” In: SPIE medical imaging, (2008).

[85] Piet Van Mieghem. “Paths in the Simple Random Graph and the Waxman Graph”. In: Probability in the Engineering and Informational Sciences, 15 (2001), pp. 535 – 555.

[86] B. Wawman. “Routing Of Multipoint Connections”. In: IEEE Journal on Selected Areas in Communications, 6.9 (1988), pp. 1617 – 1622.

[87] D. Watts, S. Strogatz. “Collective dynamics of 'small-world' networks”. In: Nature, 393 (1998), pp. 440 – 442.

[88] J. Shlens. “A Tutorial on Principal Component Analysis – Discussion and Singular Value Decomposition”. In: Princeton University (2003), pp. 1 – 16.

[89] Z. Ghahramani. “Unsupervised Learning”. In: University College London, UK (2004), pp. 1 – 32.

[90] S. Doyle, S. Agner, A. Madabhushi, M. Feldman and J. Tomaszewski. “Automated grading of breast cancer histopathology using spectral clustering with textural and architectural image features”. In: IEEE Xplore (2008), pp. 496 – 499.

[91] Y. Al-Kofahi, W. Lassoued, W. Lee and B. Roysam. “Improved automatic detection and segmentation of cell nuclei in histopathology images”. In: IEEE Transactions on Biomedical Engineering, 57.4 (2010), pp. 841 – 852.

[92] Y. Xu, J-Y. Zhu, E. Chang and Z. Tu. “Multiple Clustered Instance Learning for Histopathology Cancer Image Classification, Segmentation and Clustering”. In: Computer Vision and Pattern Recognition (Microsoft Research), (2012).

[93] R. Totha, P. Tiwaria, M. Rosenb, A. Kalyanpurc, S. Pungavkard, A. Madabhushia. “A Multi-Modal Prostate Segmentation Scheme by Combining Spectral Clustering and Active Shape Models”. In: SPIE Proceedings, 6914 (2008).

[94] D. Michie, D. Spiegelhalter, C. Taylor. “Machine Learning – Neural and Statistical Classification”. In: Overseas Press, (1994).

[95] R. Cuarana, A. Niculescu-Mizil. “An Empirical Comparison of Supervised Learning Algorithms”. In: ICML '06 Proceedings of the 23rd international conference on Machine learning, (2006), pp. 161 – 168.

[96] R. Dubes and A.K. Jain. “Clustering Techniques: A User Dilemma”. In: Pattern Recognition, Pergamon Press, 8 (1976), pp. 247 – 260.

[97] D. Svozil, V. Kvasnicka, J. Pospichal. “Introduction to multi-layer feed-forward neural networks”. In: Chemometrics and Intelligent Laboratory Systems, 39 (1997), pp. 43 – 62.

[98] D.S. Broomhead, D. Lowe. “Radial Basis Functions, Multi-Variable Functional Interpolation and Adaptive Networks”. In: Royal Signals and Radar Establishment Memorandum, 4148 (1988), pp. 1 – 31.

[99] R. Rojas. “Neural Networks – A systematic Introduction”. In: Springer, 1996.

[100] T. Kohonen. “Learning vector quantization”. In: M.A. Arbib, editor, The Handbook of Brain Theory and Neural Networks, (1995), pp. 537 – 540.

[101] J. Connor, L. Atlas. “Recurrent Neural Networks and Time Series Prediction”. In: IJCNN-91-Seattle International Joint Conference on Neural Networks, 1 (1991), pp. 301–306.

[102] A-M. Šimundić. “Measures of diagnostic accuracy: basic definitions”. In: The Journal Of The International Federation Of Clinical Chemistry And Laboratory Medicine, 19.4 (2008), pp. 1 – 9.

[103] W. Zhu, N. Zeng, N. Wang. “Sensitivity, Specificity, Accuracy, Associated Confidence Interval and ROC Analysis with Practical SAS® Implementations” In: NESUG 2010 - Health Care and Life Sciences, (2010), pp. 1 – 9.

[104] W. F. McCarthy, N. Guo. “The ROC Curve Method For The Assessment Of Diagnostic Accuracy". In:

[105] C.E. Metz. “Basic Principles of ROC Analysis”. In: Semin Nucl Med., 8.4 (1978), pp. 283 – 298.

[106] K. Hajian-Tilaki. “Receiver Operating Characteristic (ROC) Curve Analysis for Medical Diagnostic Test Evaluation”. In: Caspian J Intern Med., 4.2 (2013), pp. 627–635.

[107] M. Walker. “Diagnostic tests: ROC curves, sensitivity, and specificity”. In: Michael Walker, President of Biostatistics, Biotechnology and Bioinformatics for

Pharmaceutical and Biotechnology Research and Development, educational material.

[108] D. Aurthor and S. Vassilvitskii. “k-Means++: The Advantages of Careful Seeding”. In: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, (2007), pp. 1027 – 1035.