UiO : Department of Informatics
University of Oslo

Automatic debris detection and cell-type classification in light microscopy images of cell nuclei

Sigmund Johannes Ljosvoll Rolfsjord

15.08.2014

Abstract

Digital image analysis has proved to be a powerful tool for providing a prognosis for cancer patients. For the prognosis to be as robust and reliable as possible, information regarding cell-type is needed, and damaged or overlapping nuclei have to be removed. Manually labeling the cell nuclei is time-consuming and expensive. An automatic labeling procedure would be an important contribution to the preprocessing of cell nuclei.

In this thesis, we have developed a model for automatic classification of cell-type and removal of debris, using modern machine learning techniques. An investigation of the manual labeling of a set of experts is performed, to evaluate the performance of our approach. For removal of different types of debris we have developed highly specific novel features. We have also evaluated a set of previously known features, for use in cell-type classification.

We generally found that automatic classification can achieve similar performance to that of human experts. The best results were found to be a correct classification rate of 97 % for cell-type classification and 87 % for the complete classification of both cell-type and debris. On the same small dataset used for evaluation of the human experts we found an average correct classification rate of 79.43 %. This result was better than the worst performing human expert and within the 0.95 confidence interval ($85.14 \pm 7.29\%$).

Our approach shows promising results for automatic labeling of cell nucleus images, but may still be less robust than human experts. Further investigation of the human performance is needed to conclude on whether the whole labeling process can be fully automated and in order to chart out a direction for the further development of the automatic procedure.

Acknowledgements

This study was carried out at Institute for Cancer Genetics and Informatics at The Norwegian Radium Hospital and the Department of Informatics at the University of Oslo. It was started in January 2013 and completed in August 2014.

First I would like to thank my supervisor Professor Fritz Albregtsen for his thorough scrutiny of my thesis and for his important feedback and advice. I also would like to thank Andreas Kleppe, as I greatly appreciated his interest in my work, his advice and especially his input on detection of overlapping cells and the analysis of the inter-observer data. His thesis has also proved to be of great help throughout my work. Further I send my thanks Dr. John Maddison for reading my thesis and providing feedback. I thank my supervisor Professor Håvard E. Danielsen for providing the necessary material and data for carrying out the project, his descriptions of how the nuclei are labelled and for putting me in contact with the right people. The descriptions of the labeling process and data i received from Tarjei Sveinsgjerd Hveem and Wanja Kildal, were also much appreciated.

Finally I wish to thank my girlfriend Oda Gundersen for her patience and support.

Contents

Chapter 1

Introduction

The main aim of this study as been to develop an automatic procedure for labeling cell types and removing noise, from microscopic images of human carcinoma. These classified cells are at a later stage used to provide important prognostic information. Some cells are already automatically classified at this moment, but many cells are still manually reviewed and labelled. In fact, for each patient, thousands of cells are manually reviewed, and this is a very time-consuming procedure.

Automation of the classification procedure will not only save a substantial amount of resources, but can also provide an opportunity to analyze a larger sample of cells for each patient, which in turn may further improve the quality of the prognostic information. Despite a potentially large payoff possibly achieved by completing this task, there are very few studies available on exactly this subject. The studies that are available are mostly very restricted due to limited datasets. We are in the fortunate position of having access to a much larger dataset, which enable us to consider more complex methods, with smaller risk of overtraining.

1.1 The Project

1.1.1 An Overview of Challenges

The main challenge for this project is the relatively large uncertainty related to the data material. We have a large amount of data, but little or no information regarding the uncertainty, or the different sources of uncertainty, of this data. The data have been accumulated through many years, and we have no overview of all the possibly relevant changes throughout this period. We have no thorough investigation of the performance of the experts doing the manual classification, and no knowledge on how their performance change through time or across different data material.

Another related challenge will then be to find the most general solution achievable. The search for a general solution also has to be balanced against achieving an acceptable level of accuracy for the solution. The problem of finding this balance is further complicated, as we have no measure of what is an acceptable accuracy.

The classification task in this study could essentially be divided up in two different tasks, *filtering out* cells that are not fit for analysis and *classifying* cells based on cell type. For the first task we have better control over the different criteria that the manual classification is based on. This means that there are little hope in finding "hidden" relationships between the class and some features or interaction between features. We are essentially mimicking the judgement of expert observers, and to look for patterns in their decisions that they are not aware of, will probably be futile. For the cells-type classification on the other hand, there may exist hidden patterns in the data, that are indeed relevant. In this task our ultimate goal is not to mimic observers, but find patterns in nature. With no experience in cell biology, we have little *a priori* knowledge to incorporate into our model and are therefore forced to do a wider search through different aspects of the cell images, in hope of finding some relations between the cell images and the corresponding cell type.

With many different classes that are not always visually distinct, this may be a complex classification task. We still need to make the process somewhat transparent, so we can discover biases in the algorithm and easily adapt the approach accordingly.

1.1.2 Different Approaches

To classify a large noisy data-set, we could do a very wide search through a huge number of features, use a large training set and hope that the noise would average out. Another approach would be to develop strong features based on a priori knowledge about the problem.

Our first attempt was to scan the literature for information that we could leverage in our classification task, but this proved unrewarding. We then chose a wide-search approach for both the *filtration* and the *cell type classification* problem. We even attempted an extreme version of the wide search approach were we took the whole cell image as input and used a convolutional neural-net [51] to generate features from the image. We soon rejected this approach as it gave us no control over the errors, and no easy way of adapting the algorithm to different data-sets. We finally ended up with an approach were we focused on finding a set of features that captured as many aspects of the image as possible. We reviewed a large range of features, where some have been successfully applied to other classification tasks related to cell nuclei. Some features were excluded after investigation in the review process. As mentioned the strategy of a wide search is most relevant to the cell type classification task, but we still evaluated the same approach for the filtering as well.

For the filtering task our primary objective was to develop a set of novel features, specially designed to measure the criteria set for the cell nuclei that are to be excluded. We could then set thresholds on those measures by visual inspection. With this approach the different thresholds could be reset to adapt to different data.

We finally need some methods for evaluation of of the procedure. As we have no ground truth for the cell type classification, we could possibly evaluate the performance based on the final result on the prognostic prediction, from the later applications of the data. Such a method could be called a *goal-directed* evaluation [82]. This could be a reasonable evaluation method, at least for the exclusion of nuclei from further analysis. The exclusion process by the

human experts is more subjective in nature, and a *goal-directed* evaluation would be a way to get a more objective measure. The problem with this type of approach in our application is that our classification may be overfit to the given prognostic analysis. Since the methods of prognostic analysis is rapidly changing and improving, it would be hard to determine whether the classification method also is optimal for future methods of analysis. With this in mind we found that the best methods of evaluation would rather be to measure how well we mimic human experts. Even though the human evaluation have a certain unreliability, a result close to that of human experts, would mean that in further applications the automatic classification would have similar performance, compared to the only available alternative, namely manual classification. A natural evaluation metric is then the *correct classification rate* (CCR), which is the ratio of correct classified nuclei, in comparison to the total number of cell nuclei. A finial metric of success will then be to achieve a performance similar to that of human experts or not significantly different. With this evaluation we have to note that we cannot possibly achieve a better result than the human experts in general, but we may outperform some individuals.

To measure the performance for each class individually we use the *precision* and *recall* metrics. To evaluate classification results it is common to define the samples as *true positive*, *true negative*, *false positive* and *false negative*. Samples correctly classified to a given class are called *true positives*, while samples that does not belong to the class and are correctly classified as such are called *true negatives*. Samples that does not belong to a given class, but are classified to that class are called *false positives*, while samples that do belong to a class, but are classified to another class can be called *false negative*.

Precision for a class can then be defined as the number of true positives divided by the total number of samples classified as samples classified to the class, called *positives*. Precision is then a measure of the share of unrelated samples that are mixed into a class. Recall is the number of true positives divided by the number of samples that should belong to that class. Recall is a measure of how many samples remain in a class after classification.

Sensitivity and *specificity* are also popular measures for performance of a classification for a single class. Sensitivity in this context is the same as recall, while specificity is the number of true negatives divided by the number of samples that should be classified as negative; not part of the given class. For our current project we find it most natural to use the metric pair of precision and recall, as for each class the most relevant information is how clean the classified result is and how many samples we loose for each class. Specificity is less relevant as it primarily refers to the samples that does not belong to a class, and describe how many of those samples that are are absorbed by the class. In our multi class application it is unnatural to treat samples that do not belong to a class as an entity, as they in reality belong to different classes. This especially holds true in our application, where the distribution of samples among the classes can be very different.

Another evaluation metrics that is commonly used, if not in our study, is the *area under the curve* (AUC) which refer to the area under the *receiver operating characteristic* (ROC) curve [36]. The ROC curve describes the tradeoff between sensitivity and specificity for a classification model. It is created by plotting the sensitivity for each value of specificity. Sensitivity is usually plotted on the y-axis and $1 -$ specificity is plotted along the x-axis.

1.1.3 Our Contribution

With this study we suggest a novel set of features for identifying cells that are not segmented or prepared correctly. We also investigate a wide range of features suited for identifying different cell types and review some modern methods for classification in light of the task at hand. We identity challenges with using image analysis for classification of nuclei, and perform an investigation of possible sources of errors. Most importantly we provide an algorithm for automating a very labor intensive task.

1.1.4 Organisation

We consider chapter 1-3 to be related to what is generally call *Introduction*, in the commonly used IMRAD structure, even though some parts of chapter 2 is clearly related to *Methods*. We choose to present the material as early as possible in order give the reader some insight into cell-type classification, removal of debris and an impression of the challenges for the classification, before being exposed to previous studies. Chapter 4-7 we consider method chapters, and finally we have *Results*, *Discussion* and *Conclusion* in chapter 8-9.

Therefore, chapter 2 is a presentation of the data material, the different classes, how the data is created and labelled, and the challenges face in relation to this data. In chapter 2 we have also included a small study, we performed in order to evaluate the quality of our data material.

After describing the challenges, a natural next step is to review what prior research exists on the subject. This is important both to evaluate whether some of these finding can be used in our project and where we have to focus our resources to provided the best contribution to the field. Chapter 3 present some of the few studies that available on our exact subject. Some related to filtering debris and others related to cell-type classification, with similar classes as in the current study.

Chapter 4 is a presentation of some details related to two methods especially relevant to some of the features generated, namely ellipse fitting and Fourier descriptors. The details are included as they proved important when considering how they features could be generated. In chapter 5, we describe how we applied the methods from chapter 4 to generate a set of novel features especially developed to detect some subgroups of K6 cells. The chapter contain a description of the features and some insight into the thought processes the underlie the development of these features. Chapter 6 is a presentation of a range of features gathered from other studies, and an investigation into how they relate to the different classes of the current study.

In chapter 7 we present the classification methods we used, why these methods were both appropriate for our project in particular, and some aspects to consider when choosing and tuning a classification algorithm. In chapter 8 we present and discuss the result of our classification. Concerning both the effects of the different features, thresholding features to filter out debris and classification. Finally in chapter 9, we conclude and present our view related to further work on the subject.

Chapter 2

Material

The datasets consists of images of cell nuclei collected from biopsies of human carcinoma, collected from different patients. Throughout the thesis, these cell nuclei will often be referred to as solely "cells" or "nuclei", for simplicity.

There are many cell samples for each patient, and those cells are represented by a set of 5 images, described in section 2.5. The cells are labelled into one out of six categories, class 1 - class 6, epithelia nuclei, lymphocytes, plasma cells, stromal cells, automatically excluded nuclei and manually excluded nuclei, in the respective order. The labels K1, K2, K3, K4, K5 and K6 will be used as further reference.

We make a distinction between new and old datasets, primarily because they are captured with different cameras and therefore are significantly different. Along with the change in camera, the processing in form of pre-sorting algorithm and segmentation, have also changed.

2.1 The Datasets

We have one large set consisting of data from 80 biopsies from colorectal cancer, named M51. The training set comprised of 140533 cells, with 56.64% K1, 0.44% K2, 15.82% (K3), 0.63% K4 and 26.46% K6 cells. The M51 test set had a total of 17669 cells, with 61.82% K1, 0.18% K2, 11.11% K3, 0.58% K4 and 26.31% K6 cells. We also had a data-set from breast cancer tissue, named L41, from a total of 60 patients. In our training-set for L41 we had a total of 75116 cells, with 61.87% K1, 1.49% K2, 6.69% K3, 0.69% K4 and 29.25% K6 cells. The test-set had 13038 cells with 66.56% K1, 1.93% K2, 3.91% K3, 0.29% K4 and 27.31% K6 cells. Finally we had a somewhat smaller data-set from cervical cancer from 30 patients, called PLM13. Our training-set had 33221 cells, with 66.32% K1, 0.80% K2, 4.57% K3, 0.65% K4 and 27.86% K6 cells. The test-set had 24433 cells, with 64.44% K1, 0.67% K2, 4.97%K3, 1.31% K4 and 28.60% K6 cells. From the PLM13 data-set, we also have cell-samples from one patient that have been classified by 7 different experts, with a total of 2989 cells, were 52.59% of those cells belong to K1, 8.40% to K2, 21.24% to K3, 0.37% to K4 and 520 cells belong to K6. With "belong" we mean what the samples are classified as the official label, used for further analysis.

We also had two sets P02 and P14 from prostate cancer and cervical cancer,

with 10 patients in each set. These sets were primarily used for developing the features for removing debris. Since the sets were small, and we had no test-set, we did not use these sets to evaluate our results. We also believed that since the features were developed primarily on these sets, we would experience a sort of overtraining, using these datasets.

The L41 and M51 are quite old sets, but some of the data have been revised later in time. There have also been used different segmentation technology and so on thorough the generation of the data. The PLM13 data-set is newer and were generated in 2013. Even though there have been used many different expert labelers, we have to remember that the labeling of this set may have occurred closer in time and the segmentation technology have probably change less through just one year, compared to the many years M51 and L41 have been in use.

2.2 Preparation of Cell Nuclei Images

All the material consists of isolated nuclei, called monolayers. The nuclei were extracted and prepared, from paraffin-embedded tissue fixed in 4% buffered formalin, using a modification of Hedley's method [40] and Feulgen-Schiff stained according to the protocol described in [80]. A Zeiss Axioplan microscope with a 40/0.75 objective lens, a 546 nm green filter was used in the process of capturing the images. For M51 and L41 we used a high-resolution digital camera (C4742-95, Hamamatsu Photonics K.K., Hamamatsu, Japan) with 1024×1024 pixels/image and a gray-level resolution of 10 bits/pixel was used to capture the images. One pixel corresponded to 166 nm on the cell specimen [62]. For PLM14, PLM14 and P02, we used a high-resolution digital camera (Axiocam MRM, Zeiss) with 1040×1388 pixel/image and a gray-level resolution of 12 bits. The gray-levels was later reduced to 10 bits, by removing the two least significants bits. The images were shading corrected by dividing each frame by an image of the background, without nuclei.

2.3 Segmentation and Sorting

The images were automatically segmented, but we have no information in regard to what method was used at given at time, and only know that they have been refined and updated many times throughout the years of the data generation. The images were first automatically sorted by a rule based classification system, as a part of the Ploidy Work Station (Room4, Crowborough, UK). We only had access to some detail on this system towards the end of the project, and the details regarding the current system of pre-sorting have not accessible. Therefore this system is not thoroughly discussed in this Thesis. Still we include some features proposed by Maddison, in an early version of the pre-sorting system, described in [54].

2.4 The Different Classes

As we have no *a priori* information regarding the cell labels, we cannot provide a thorough introduction to the different aspects of the cell classes. We also

found much of the information we did acquire to be of little or no importance for our project. Therefore we only provide a very superficial introduction of the different classes and some of the characteristics of the classes that we found to be consistent through the different datasets.

Along with the descriptions of the different classes, there is also included a set of images of cells from the corresponding class. The selection of images is aimed both to present characteristic cells of each class and to illustrated the normal variation in that class.

2.4.1 Class 1 - Epithelial Cells

As our data-set is collected from carcinoma, it is the epithelial cells that are affected by cancer. In other words this is the most important group for further classification. As cancer cells are recognized by rapid division and wild mutations, this group is quite heterogeneous and they have many outliers in regard to most features. There is huge variations in shape, size and texture. This makes the classification task more complex than detecting healthy epithelial tissue.

Generally the epithelial cells are bigger than plasma cells (K3) and lymphocytes (K2). Compared to the stroma cells (K4), epithelial cells usually have a more circular shape and a more grainy pattern, but there are not always a clear distinction between the two classes. Examples of K1 cells can be found in figure 2.1.

2.4.2 Class 2 - Lymphocytes

Lymphocytes cells are small compared to the other classes. They are also very dark, as the DNA in the nuclei is concentrated in over a smaller area. Additionally they are mostly very circular in shape. The fact that they also are easier to segment, as they have a strong contrast to the background, also attributes to their smooth contour. These cells does not have a clear distinction relative to many cells in K6, as they could also be debris from apoptotic nuclei that often occur in cancer tissue, and should be in K6 for our purpose. A problem related to the data material is that we have very few samples of this cell type. Examples of K2 cells can be found in figure 2.2.

2.4.3 Class 3 - Plasma cells

Plasma cells are mostly smaller than epithelial cells and have a quite circular shape, but it is sometimes hard to differentiate between large plasma cells and small epithelial cells. Some have a quite grainy texture and some are more blurred. Often it is the most blurred plasma cells that get sorted out and put in to K6, but the distinction is not quite clear here either. Examples of K5 cells can be found in figure 2.3.

2.4.4 Class 4 - Stromal Cells

They are most recognizable for their oblong shape, and some also have a dark stripe along the middle of the cell. They tend to have a light color, but some cells have a more grainy pattern similar to that of epithelial cells. We find that

stromal cells and oblong epithelial cell are often confused. Examples of K4 cells can be found in figure 2.4.

2.4.5 Class 5 - Automatically Excluded Nuclei

For the old datasets, there are generally no cells in this class. The cells that would otherwise belong to this class are removed as part of the segmentation process. For the cells in the old datasets, this class was used as a default label, before the manual classification. For the new datasets on the other and hand, the cells are presorted, and the cells that are automatically removed are kept in class 5. The images in this category are usually cell images that obviously needs to be removed. Typically very small objects or large clusters of cells. We consider this first filtering a *solved problem*, as this group is mostly ignored through a manual classification. For this project we consider the removal of these K5 cells to be a part of the segmentation process, and the class is therefore excluded from our study entirely. Still we can easily see that there are some mistakes, typically for cells with irregular shape. Still it would be hard for us to evaluate these mistakes, as the cells are still kept in K5 after manual classification. Examples of K5 cells can be found in figure 2.5.

2.4.6 Class 6 - Excluded Nuclei

This group consists of cells that should not be used for further analysis and are therefore manually removed. In this class there are:

1. Cut nuclei
2. Nuclei with damaged cell membrane
3. Overlapping nuclei
4. Nuclei with foreign objects
5. Nuclei that are badly over or under segmented
6. Nuclei that appear blurred in the image

As K6 nuclei can come from all of the 4 different cell types, this group is indeed very heterogeneous. It is also overlapping on most features with all of the other classes. For that reason it is very difficult to find a typical K6 cell. Examples of K6 cells can be found in figure 2.6.

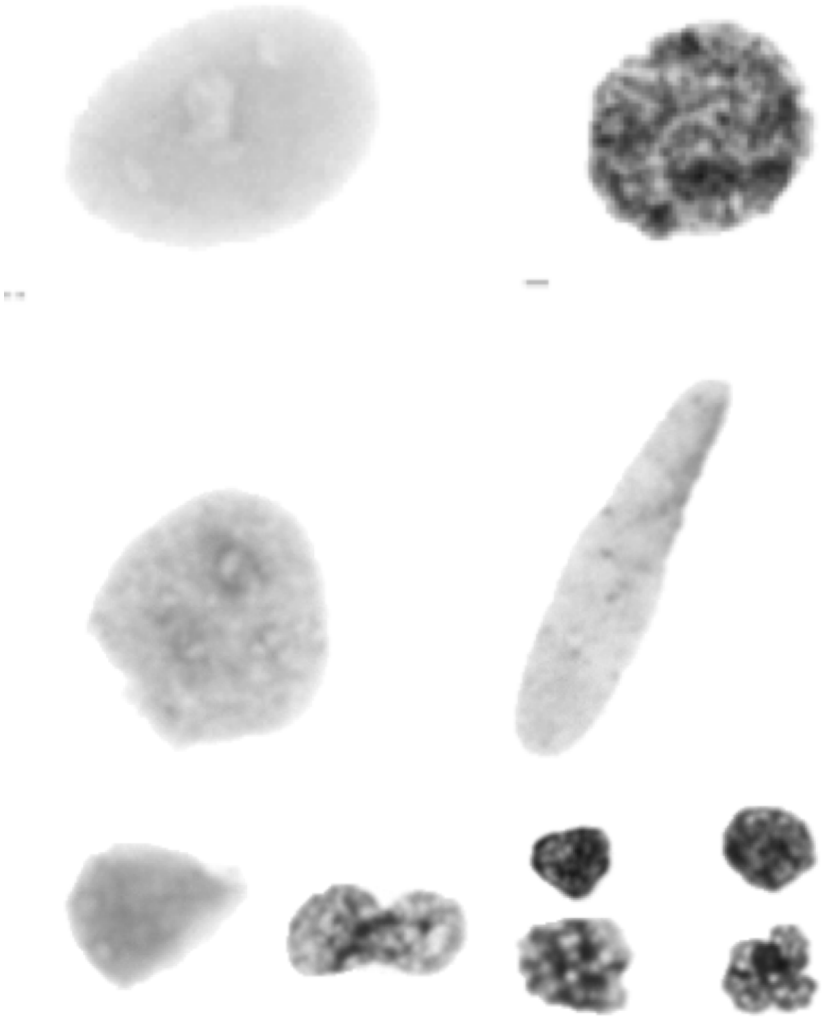


Figure 2.1: A range of different K1 cells, from the PLM13 training-set. 1 cm in the image correspond to 5.4 μm on the specimen.

Figure 2.2: K2 cells from the PLM13 training-set. 1 cm in the image correspond to 5.4 μm on the specimen.

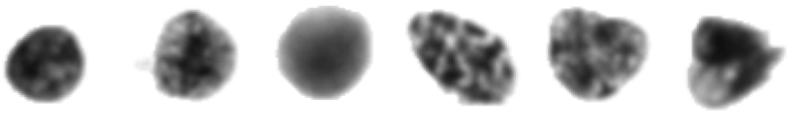


Figure 2.3: K3 cells from the PLM13 training-set. 1 cm in the image correspond to 5.4 μm on the specimen.

Figure 2.4: K4 cells from the PLM13 training-set. 1 cm in the image correspond to 5.4 μm on the specimen.

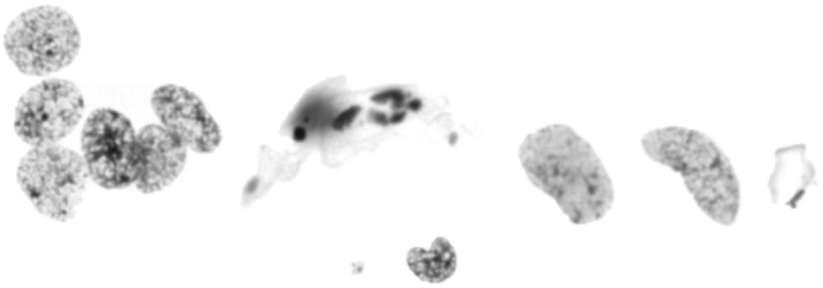


Figure 2.5: K5 images from the PLM13 training-set. These cells are scale to half the size compared to the images for the other classes. 1 cm in the image correspond to 2.7 μm on the specimen.

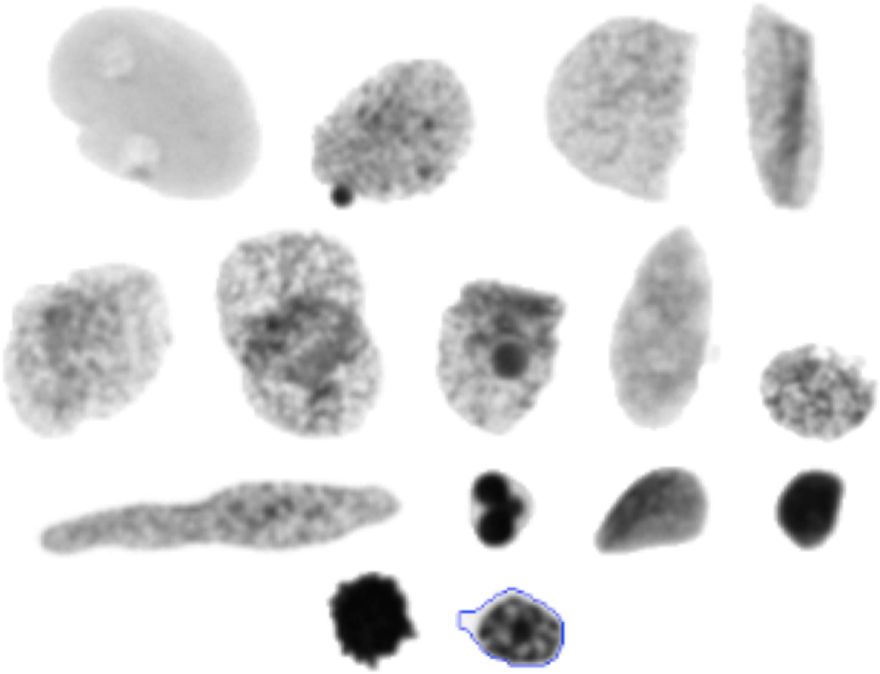


Figure 2.6: K6 cells from the PLM13 training-set. On the bottom right cell the contour has been has been outlined, to illustrate over-segmentation. 1 cm in the image correspond to 5.4 μm on the specimen.

2.5 Cell Images

For each cell we have 5 different images. We have the *original*, a *shade corrected*, a *shade corrected and segmented* and a *background* image, additionally we have a *mask* image which is a binary image, created by an automatic segmentation algorithm. The region where the binary image is *true*, corresponds to the cell area found by the segmentation algorithm and is drawn in white.

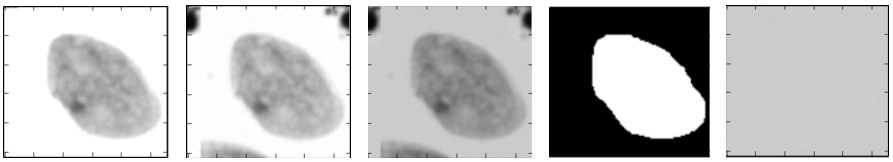


Figure 2.7: The leftmost image is shading corrected and segmented, next we have the shading corrected image, and in the center we have the original image. The forth image from the left is the cell mask and finally rightmost we have the background image. The "ticks" along the frame each represent 20 pixels.

2.6 Study: Determine Inter-Observer Reliability

The PLM13 dataset was classified by several different experts. Since the process was mainly done with visual inspection, we had to assume a certain degree of inter-observer disagreement. To evaluate the performance of the computer algorithm we had to at least have some estimate of this unreliability of the class labels and at the start of this project we had only anecdotal evidence that the reliability of the data was "very high". As mentioned the set was first and foremost classified to gain prognostic information. Therefore a main focus was to classify enough cells, to get a reliable result. We believed the inter-observer reliability to be one of the most important factors for discrepancies in the data. It is also easier to investigate than the how the datasets change through time.

If we were to investigate to what degree there were any significant differences among the expert we would need the experts to label the same set at least twice each, so we also could get an estimate of the variation for each individual. This information is unavailable to us, but for our project this does not have much relevance. We must presume that the differences in labeling is in part due to individual differences and variance in personal performance. Since we have no information about who labelled our set we are more interested in the general reliability in the data, which we can find as the sum of the two components. In other words the results we find will be this sum of individual differences and individual variance.

A question that may be of interest is whether there are some specific parts of the data that are disputable or if all parts of the data are prone to error, which may indicate that the discrepancies are mainly due to random errors.

To investigate these questions we run two different test. First we evaluate the pairwise agreement among the observers and present an average confusion matrix, indicating what mistakes that are the most common. Then we investigate how the mutual agreement decreases when we combine more observers. If we have a rapid decline of accuracy as a function of the number of observers we

would expect the errors to be more random in nature, and that each expert has some errors regardless of the data. We have to assume that there is a certain, smaller part of the data that is inherently hard to classify, if the accuracies on the other hand do not change much.

2.6.1 Methods

To investigate this we used 7 experts all labeling the same set of 8941 cell images from one patient in the PLM13 material, where 5952 (67 %) belonged to class 5. The set was first sorted by a rough rule-based algorithm, then reviewed and "correcte" by the experts. All the experts were informed of the circumstances of this test prior to their classification work. We found that the largest class, namely class 5, with the obviously corrupted cell images, was rarely reviewed at all, but rather trusted to the computer algorithm. Only a total of 15 cells were actually moved from this class and there seemed to be no agreement of which cell that should be taken out from this group. Such a large number of cells not reviewed or moved, would exaggerate the reliability enormously. We therefore excluded class 5 from our study and from further investigation. The average marginal probabilities for the classes left was: 0.513 (K1), 0.098 (K2), 0.165 (K3), 0.005 (K4) and 0.219 (K6).

Estimating Pairwise Reliability

We estimated the reliability for all possible combinations of pairs, and averaged the results. We will also report maximum and minimum agreement and the confidence interval of the reliability scores. We have to remember that that the reliability scores, simply calculated as the correct classification rate (CCR), will be somewhat biased by the uneven distribution among the classes. We will therefore also report the Cohen's kappa coefficient [23], which is a measure that corrects for such an uneven distribution. This measure assumes that the marginal probabilities for the classes are fixed [9]. This is not necessarily true, as they could vary for different types of cancers and among individuals, but it is at least true within a certain degree. The Cohen's kappa coefficient can be calculated as

$$\kappa = \frac{Pr(a) - Pr(e)}{1 - Pr(e)}, \tag{2.1}$$

where $Pr(a)$ is the observed agreement and $Pr(e)$ is the probability that the observers would agree on a samples by chance, taking the a priori probabilities for each class into account.

Mutual Agreement

To investigate the mutual agreement between a larger number of observers, we plot the probability of a given number of observers to agree on a sample. To find this probability we simply calculate the average agreement among all possible combinations of observers. The number of combinations is $\frac{N!}{k!(N-k)!}$, where N is the total number of observers and k is the number of observers to choose. This means for example that for two observers we have 21 possible combinations to average from, for three observers we have 35 combinations, while for 7 observers there is only one possible combination, namely using all observers. We do

this same procedure for each class as well. Then we calculate the ratio as the number of cells in the class that all the observers agreed upon divided by the total number of cells that at least one of the observers label to that class.

Improvement Over Pre-Sorting

In order to investigate whether the agreement is due to either that the pre-sorting is good, an anchoring effect or a real underlying agreement between the observers, we measure to what degree the observers actually change the pre-sorted labels, and if they agree on the label changes. This investigation may not provide a definite answer as we do not know, for example if the agreement between human and computer is due to the anchoring effect or actually agreement. In order to test this, the observers would have to do the labeling without any prior sorting.

For this investigation the samples where an expert agrees with the pre-sorting will be removed. In other words we will only investigate the samples that each of the expert relabeled. Then we will again present an average confusion matrix, estimated from all possible combinations of observer pairs. We also present the average agreement of relabeling to different the different classes.

2.6.2 Results

The average pairwise reliability was ($85.14 \pm 7.29\%$), with a Cohen's kappa of 0.772, which is inside what Landis and Koch [50] judge to be *substantial* agreement. The highest agreement between any pair of observers was 91.0%, and the lowest 76.9%. We found that the average reliability for separating between K6 and the other classes is 86.86 %, which gives a Cohen's kappa of 0.607. This kappa is just slightly outside the range deemed as substantial agreement and in the range of *moderate* agreement.

Table 2.1: The averaged confusion matrix from pairwise comparison of class labeling.

		Observer 2				
		K1	K2	K3	K4	K6
Observer 1	K1	1421.4	0.0	0.1	6.5	105.9
	K2	0.0	230.8	19.0	0.0	42.1
	K3	0.1	19.0	430.8	0.0	44.3
	K4	6.5	0.0	0.0	4.3	4.2
	K6	105.9	42.1	44.3	4.2	457.6

Average CCR between the observers: 85.14%

What does seem sightly worrying is that a pair of observers, on average, only agreed upon 53.79 % of the cells they wanted to label as K6 and thereby remove from further analysis. Disregarding the cells that were classified as K6 by at least one of the observers, we are left with an inter-observer agreement of 97.59 %, which gives a Cohen's kappa of 0.953, where the coefficient is in the range judged as *almost perfect* agreement.

From the plots in figure 2.8 we can see that the agreement decreases quite rapidly and that this decline differ significantly between the different classes.

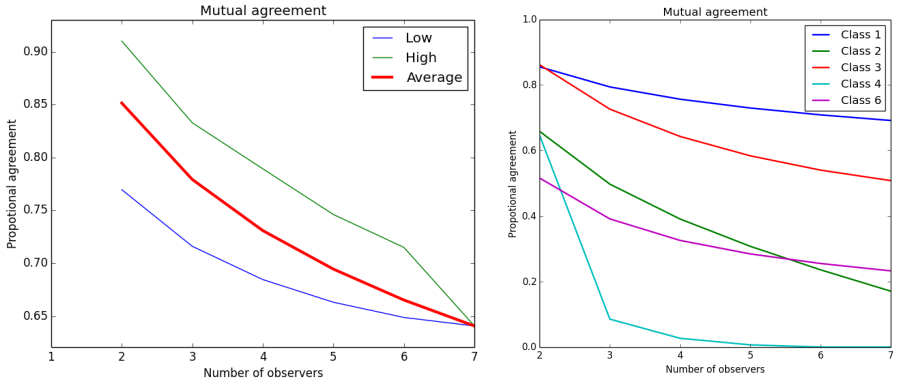


Figure 2.8: The leftmost plot show how the mutual agreement decrease for each added observer. The green line represents the decrease when the subset of observers with the highest possible mutual agreement are chosen for each step. The blue line show the decrease when the subset of observers with the lowest mutual agreement are chosen for each step. The red line show the average mutual agreement between all possible subset of observer with a given size. The lines meet at 7 observers since there are only 7 observers in total, and there is only one possible set of observers for that size. The rightmost plot shows this average agreement for each of the individual classes.

The observers all agreed upon in total 1214 K1 cells, 65 K2 cells, 342 K3 cells, 0 K4 cells and 294 K6 cells.

In table 2.2, the confusion matrix for relabeling is presented. On average about 30% of the cells where moved, but a pair of experts only agree on about half of those relabelings. Of all the relabeling, 94% was to the K6 class. This means that at the current development point in time, the primary focus of the expert is to remove additional debris. From table 2.3 it is also clear that K6 is the class with the highest agreement on the relabeling.

Table 2.2: The averaged confusion matrix from pairwise comparison of class relabeling. The samples under the label "Pre" was not relabeled from the original pre-sorting.

	K1	K2	K3	K4	K6	Pre
K1	4.52	0.	0.	0.	1.69	3.64
K2	0.	1.71	0.	0.	0.95	0.19
K3	0.	0.	1.86	0.	2.1	18.76
K4	0.	0.	0.	0.38	0.24	2.95
K6	1.69	0.95	2.1	0.24	457.62	191.55
Pre	3.64	0.19	18.76	2.95	191.55	2078.76

The average total agreement of relabeled samples was 51.21%

Table 2.3: The percentage of relabeling agreement for each class.

K1	29.78%
K2	42.86%
K3	4.26%
K4	5.63%
K6	53.80%

2.6.3 Discussion

From our results it seems that the experts agree on large parts of the K1-K4 cells, while the K6 cells are more disputed. We still have to remember that there are several aspects that may affect the result. We may question whether the fact that the experts knew they were tested, can have an effect on the result. It may at least be that some arbitrary mistakes are moved as they may be more thorough on a test compared to a normal working situation. Secondly we do not know how much they were influenced by the original sorting of the cells. In this study we are not able to measure the effect of this as they all used the same pre-sorting.

The highest achieving result seems very good, but unfortunately the fact that two experts are in quite high agreement does not necessarily prove very helpful to us. This is because we will end up with a model trained on a set, labelled by many different experts and therefore we will come closer to the average. What is more disturbing is the the lowest agreement of only 76.9%. As we test our algorithm on comparably smaller data-sets we may end up comparing our result with only a small number of experts. This means that we can experience great variance in our test results, purely due to the performance of these experts.

As we might expect, the error seems to stem partly from more arbitrary mistakes and that there are some parts of the data that are difficult to label. Even though the plots in either case are expected to flatten out, we can see that there are differences between the classes. A part of this seems to be related to the relative number of cells in each class. We can imagine for example that some of the K4 cells are "lost in the crowd", with a comparably small number of cells it may be overlooked in the sorting process. This may at least be part of the reason for the substantial drop of the K4 class. In the same way we can also see that the larger classes have a less steep decline. It may of course also be that K2 cells, looking quite similar, are more arbitrarily classified as K6 cell, than say K1 cells. If the errors of K6 were purely due to random mistakes, these mistakes would have to be done in about 26.2 % of the samples, if we calculate this from the accuracy of all the pairs. This would lead to a final agreement of all 7 experts on 10.8% of the samples. We see from our plot that this is not the case. The final agreement between the observers for this class is 23.3%, which leads us to believe that they are more confident on some parts of the data than on others, which also seem quite obvious. The problem is that the uncertainty apparently applies to a large part of the set, perhaps up to 80%. When we have an inherent uncertainty related to a large part of the data it will be harder to identify which labels that are uncertain and which are not.

When we investigate which samples is actually relabeled it becomes evident that the only significant relabelings are those labelled to K6. With this investi-

gation we cannot know if the apparent agreement for the K1-K1 classes is due to an anchoring effect or not, but it seem unlikely that the somewhat arbitrary relabeling of the a small number of nuclei can affect the outcome of a further analysis.

2.7 Further Use of the Classified Cell Images

For our application the final cell classification is primarily used to provide prognostic information concerning a patient. It has been show that both ploidy and texture data from the cell images can provide such information. In a study [48] by Kristensen et al. found that they could differentiate the patient into different groups with different rate of 10-year relapse-free survival, based on the histograms of cell ploidy, obtained from cell images. In a long range of studies, it have also been shown that cell texture from microscopy images can also provide such prognostic information [61, 62, 63, 64]. Through years of studies, a wide range of different texture information have been used, so an overview would demand a large chapter in this thesis, but a review of the subject is already written by Nielsen et al. [61].

Even though this is slightly outside the field of our study, we still need some information regarding the use of our final result. At the start of our project we were urged not to use information related to the information used in this prognostic applications of the data. This is understandable as a classification based on this information could create a bias, that could corrupt further analysis. Unfortunately, as we find that this information is very extensive, that challenge could prove impossible.

2.7.1 Cell Ploidy

We can divide cell nuclei into different groups based on their DNA content. Cancer cells with a runaway cell growth, can often have a higher than normal amount of DNA. Cells that contain a normal amount of DNA, with 46 chromosomes, are called *diploid*. *Tetraploid* cells contain double the amount of normal DNA. Normal cells can also contain this level of DNA at at the *anaphase* of the cell division process, before the cell divides. Cells called *octaploid* and *hexadecaploid* have four and eight times the normal DNA content. These cells with a factor of a positive power of two are also called *euploid* cells.

A tumor could then be classified based on a histogram of the estimated DNA content in a cell. The DNA content was estimated based on *integrated optical density* IOD, calculated from the cell images.

In the study of Kristensen et al. [48] they divided the tumors into 4 different groups. A tumor was labelled as diploid if there was only one peak in the histogram at the normal level of DNA, while the number of cells in the tetraploid area was less than 10% of the total number of cells. A tumor was defined as tetraploid either if the cells in the tetraploid region exceeded 10% of the total number of cells, or there were both a peak in the tetraploid and the octaploid regions. A polyploid tumor had both a peak of octaploid cells and a peak of hexadecaploid cells. Finally a tumor could be classified as aneuploid when the histogram had peaks outside the euploid areas or when the number of cells with

DNA content higher than 2.5 times the a normal cell, that were not part of an euploid region, was higher than 1%.

2.8 Challenges with the Material

If we use a supervised learning strategy, there is no way of getting a better result than the reliability of our training data, although a large training-set could average out the individual differences of each expert. As I have no education in pathology or molecular biology, I am essentially bound to the training data. This means that the materials have the uttermost importance in this project.

2.8.1 Class Lables

The classes K1, K2, K3 and K4 are biologically different. This means that for these classes there exist an absolute ground truth. From our inter-observer reliability study described in 2.6, we can also see that there is a relativity high agreement for these classes as well. The observers mostly agree upon which are epithelial cells and which are not. It might be possible to get a result even closer to the ground truth by using staining procedures, but this type of information is not available to us for this project. In our study we saw that we have no certainty regarding the class labeling. In other words we can not get a good estimate of the true accuracy. If we use a supervised learning strategy, there is also no way of getting a better result than the reliability of our training labels. In our reliability study we saw that the classification of the K6 cells seemed to be somewhat subjective. A major challenge is then to evaluate which mislabeling can be ignored and how we can make sure that some mislabeled cells will not corrupt the classification of other cells.

It is not only the effect of multiple observers that is a concern for the quality of data material. As the data has been collected over a long time period the technologies have also changed considerably. Additionally there may be a development of the observers through time, where they change their focus to different features of the cells. There could also be a tendency that some sets are classified more thoroughly than others.

The unreliability of multiple observers is a concern, but each individual expert may also change opinion with time and training, and may be uncertain in regard to some cells, and therefore label the same cells different at different points in time. It may be that some sets are more thoroughly classified than other, depending on the further application that they were primarily intended for.

Not only do the experts change through time, the technology change as well, and this may have an effect on the class labels. On the newer data-sets the cells are first roughly sorted by an automatic algorithm. This first automatic classification can also have an effect on the manual classification as the observers can be influenced by the initial labels, as Jacowits and Kahneman have showed [46]. As the automatic classification procedure have changed drastically for the material it could possibly result in large discrepancies in the data. This is further complicated by the fact that we have no information regarding the automatic classification procedure used at the time of the labeling. The anchoring factor

of the manual classification should ideally have been investigated to get a better estimate of the reliability of the data.

In our opinion much of the unreliability in of data are caused by the fact that the data collection have been dominated by an emphasis on quantity. This makes sense since the data were originally gathered to develop algorithms for prognostic information. For prognostic information each patient is treated as one sample, in comparison to each cell image being treated as a sample in our case. This means that the need for training data is much higher than in our case and they had to focus on quantity to simply get enough data. In their case the uncertainty of individual cells would hopefully be averaged out for each patient. In our case on the other hand, we are as mentioned in a situation where we cannot get a better result than the training data and most probably our result will be worse. This means that a good strategy for improving our automatic classification could be to develop a training set based on quality, but solving this challenge is not up to us.

2.8.2 Differences in the Cell Images

Perhaps the main goal of a classification task is to find features or relationships between features that provide information and thereby predict the class label. Those relationships between the class label and the features also have to be stable throughout the population of possible samples. In other words, we have to search for traits of the different classes that remain the *stable* for all the data we want to generalize our approach to. The problem is that it is difficult, if even possible, to predict what systematic changes may occur in future data, that we have not investigated. Typical systematic changes may for example occur with changes in the technology.

Throughout the data collection period there have been used multiple algorithms for image segmentation. The different algorithms have resulted in different types of artifacts. Some segmentation algorithms give very jagged edges, some tend to over-segment while others tend to under-segment the cells. This may have a drastic effect on the population of K6 cells. If the boundaries of accepted cells had not changed in relation to the segmentation algorithm we would have no concerns. The problem is that the manual labeling is a very subjective procedure. If for example all of the cells have a very jagged edge, it would make no sense to exclude all the cells for that patient. Therefore the manual classification depends on the quality of the cell images for each individual patient. It is of course possible to do exactly the same thing for the automatic classification, using features relative to the other images from a patient. Logically on the other hand this makes little sense, as if one cell is too blurred to be used in further analysis for one patient, it should also be too blurred for other patients, but perhaps that some images are better than no images.

The imaging procedure have also changed throughout the collection period. First and foremost by changing the microscope, which gave higher resolution and higher contrast. This means that the cells seem relatively larger and the coloring and contrast is different. In practice this means that at least the thresholds are not generalizable from before and after the change of microscope. It has also been some changes in staining procedures, that for example can result in slightly different coloring and patterns.

These differences in the image are something that either have to be standardized or we have to average them out over may different training samples. There are still differences between the cell images, patients and data-sets that we cannot easily "escape" by some standardization of our methods. Differences that we have to average out through a large training-set.

Cancer tissue located in different regions often have different qualities. Cells from cervical cancer may for example look different than cells from colorectal cancer. There may also be individual differences between patients; each patient may also of course have individual differences. The individual differences is exactly what results in important prognostic information. The problem is that we can not simply compare the data-sets to get a measure of reliability and generalizability as we cannot know if these differences stems from differences between cancer types, individual differences, different technique or different observers. If we had the information about who the observer were for each image and what techniques were used, we would have a better opportunity to derive a better measure of the reliability of the data.

2.8.3 Creating Bias in Later Applications

As we mentioned in the section on further use of the cell images, section 2.7, we were urged to not use information that were utilized in the final applications of the sorted data-set. This basically relates to information about the gray-level histograms and the texture information. It may be that some information concerning the gray-level histograms could be used, as the information in the further application on this subject is primarily IOD, but perhaps also entropy. IOD consist primarily of information on the mean intensity and area, but it is a more robust measure for the DNA content as it both accounts for differences in the background, and disregards over-segmented areas completely. If we include information of both mean intensity and cell area we will capture much of this information, but not all.

The main problem in our application is that overlapping cells have very high levels of IOD, obviously because they contain the DNA of two or more cells, instead of one. When we see that a prognosis can be decided on the basis of merely 1% of these high IOD cells, we understand that it may be vulnerable to such overlapping cells. If we had more overlapping cells in the training set than these high IOD cells, and these cells were not substantially separated on other measures, we would end up in a situation where the high IOD were excluded as overlapping, and this could severely affect the prognosis. The problem is reenforced by the fact that these high IOD cells exists only in a few patients, and the overlapping cells will mainly affect the regions indicating polyploid tumors, as the combination of two cells will of course exactly double the DNA. We can se that from the study of Kristensen et al. [48], they only classified 10 tumors as polyploid out of 284 samples in total. Since all samples will usually contain some overlapping cells, these high IOD cells may very well be outnumbered.

Cut cells will probably not have the same influence. Normally cut cell will have abnormally low IOD as some of the DNA content is lost in the cut. That means that these cells, at least for ploidy analysis, will not affect the final prognosis. Still if many high IOD cells are cut, they can end up with an IOD outside of the euploid peaks, as the remaining DNA after a cut is rarely a factor of 2^n of the normal DNA content. This means that patients in the polyploid

prognostic group could be classified as aneuploid. As these are more rare it is less likely that this will affect the result in any major way.

The effect of using texture information is less clear. Since the K6 cells represent all groups and texture is not directly one of the criteria for removing cells, we would not expect to see systematic differences in texture between the K6 cells an other cell. Still we can easily imagine that overlapping cells or cell with foreign objects will have an effect on the texture measures. Additionally blur is mentioned as a criteria for removing cells, and this is obviously related to texture. Still for affecting the prognosis we believed that texture is more of an issue in the cell-type classification. It may be that the "less dangerous" cancer cells have texture more similar to K3 or K4 cells and we could end up classifying these diploid cancer cells into one of these categories, and thereby remove them from the correct analysis. Then the part of the most severe cancer cells will seem relatively larger. This effect can also be viewed in the way where a thresholding on one of the texture features will cap the range of this parameter for further analysis and therefore skew the population, which again would probably affect the result of classification on those features at a later time.

The problem with the texture measures and especially combinations of texture measures is that it may be very hard to determine what they actually measure. Especially with the adaptive texture measures, as they are essentially a linear combination of a huge range of features. This is further complicated by the fact that such a huge range of texture measures have been tested. One can easily see that for example adaptive gray-level entropy features [62] also can provide information on the general entropy of a cell. In the same way many of the other texture measures can be affected by the general gray-level histogram of a cell.

To be absolutely sure not to affect the result in any such way we should in other words not use texture or first order gray-level information. Then the only possible information left is morphological information. Unfortunately we found this information to be very unreliable, as the segmentation algorithm change, both the shape of the contour and the size of the cells change dramatically, also the experts view on what level of jaggedness is acceptable changes dramatically. We find that almost no samples are separable purely based on morphological information and even overlapping cells are far from separable without texture information, even for a trained human eye.

In stead of restricting our information we use the radically opposite approach, and hold no information as sacred. If we were to include some gray-level information, but not IOD, the IOD of a cell would probably still have a huge effect on the classification, as this is a very strong feature and the classifier combines the information in many ways. As it is hard to interpret this combinations of features we find it better to include the IOD information directly, so we at least can study the impact of the feature directly. The IOD features could also have a positive effect, as many of the high IOD cells have very rough edges, but the high IOD balance this out in a way so the cells are still kept. The effect of the texture features is hard to study regardless of how we apply them, but we believe that the result will probably be less skewed by finding a result as close to the human experts as possible, rather than a worse result that is unaffected by the texture information.

2.9 Training and Test

2.9.1 Training-Sets

The training-set will determine the end result and form the decision boundary. In that sense one could say that choosing the training-set is one of the most important decisions in the classification process. An ideal training-set is both accurate and representative for the data we want to generalize the model to. As is very common in machine learning tasks, we have little knowledge related to what is accurate and representative, so in our case this selection process is quite simple. What we do know is that to get a representative training-set we need at least some cells from different prognostic groups. So we do make sure that we have include patients from all ploidy groups, as with an unbalance in this regard we could end up with some prognostic groups being more susceptible to errors than other, which again may lead to major biases in later analysis.

The best possible solution would be to have one large training-set for all the different data-sets. Unfortunately we find that the different sets differ to such a degree that we do think it is best to train different training-sets for each material in order to get an acceptable accuracy. This do mean that we will have to train a new model for each major change in the material, which is time consuming, but perhaps necessary.

One of the main problems with the data-set is the large difference of cell-samples for each class. In the whole set of M51 there are 613 nuclei from K2, 881 from K4 and 79542 from K1. With a classification process that optimize the CCR, small groups of cells will be down-prioritized. In the inter-observer study we learned that also the experts are influenced by this effect. From Table 2.2 we can se that on average only 16.73 % of the cells in K4 were undisputed on average between two observers. For K2 on the other hand 65.38 % of the cells were undisputed. By creating a uniform training set we are essentially valuing each class equally; labeling 2 % of the K2 nuclei as K1 is just as bad as labeling 2 % of the K1 nuclei as K2. This means that for each patient the total CCR actually goes down, since 2 % from K1 is a lot more than 2 % from K2. To get an accurate evaluation of the classification for each class it would be most natural to keep both an equal training-set and a test-set, but to mimic the observers it might be most reasonable to keep the data as it is. If we were to balance out the training data we would end up with very small data-sets, specially for the K1 and K6 class as they are very heterogeneous groups.

We found that for the cell type classification we will try to even out the data sets, as we are trying to se how well we can classify the different cells. Wen we classify the whole data set on the other hand we will use the whole data sets, both because the K6 cells are very heterogeneous and need many examples, but in that part of our testing process we want to see how well a computer can mimic the human experts, as a test to whether we could replace the entire manual classification procedure.

2.9.2 Independent Test-Set

When we evaluate our algorithm, we want to measure how the classifier is going to preform when we have no information regarding the labeling of a data-set. It is quite obvious that training and testing on the same data-set will give

optimistic results. The training fits the model to the data-set, and if the model is complex enough it can be perfectly fit to the training data. It is however easy to forget that *feature selection*, *grid search* and *model selection* also is a way to fit the model to the training set. Cross-validation is often used to evaluate the progression in training, but if this cross-validation result is further used to optimize the model, a independent test set needs to be held back. Schulerud showed with a simulation, how feature selection in combination with cross-validation could lead to dramatic overestimation of the model accuracy , when she did not use an independent test set [75].

We also have to be aware of any other way of adapting the model to the test set. Through a long process of developing a classification algorithm, parameters, scaling and normalization may be chosen in such a way that it fit the training set. To get an unbiased estimation of the model accuracy we keep one test set for each training set, untouched until the algorithm is fully developed. Each of the testing-sets consists of a series of cell from 10 patients. In [61], they discuss how to optimally divide the data into training- and test-set, as a small training-set can give a larger classification error, while a small test-set can give high variance in the error estimation. For the current project this issue is of less importance, as our test set was first obtained after the development of our procedure. The number of samples in each set is chosen by the number of patients and not by the number of sample images. This division is in other words not based on an optimal selection based on true error rate or dimensionality our model. In the study [61], they do indicate that when the data have a large true error, it should be used a large test set, while high dimensionality model should have larger training-sets. For the current project this might indicate that we need quite large test-sets to achieve reliable results, as the class overlap is quite high.

2.10 How general can we make our model?

The main reason for creating a general model is to avoid creating multiple models for different scenarios. To use a different model for each tissue-type, for example, can be unpractical and perhaps make room for error. We might also expect that a model that works for different tissue-types will also be more robust for chances within the tissue-types, as well. The main problem of creating such a general model is the constant changes and improvement in the preparation of the image slides and segmentation algorithms. We find that the datasets, differ substantially, based on when they were generated. Although we find that the main differences between the datasets can be attributed to changes in time, we cannot completely rule out differences based on tissue-type. This is because the data we have for the different tissue-types also are generated at different times.

If a model should be robust for large changes in both gray-level, shape and size, we would loose much of the differences we base our classification on. In outer words, we would have to expect a decrease in performance, if we were to create a very general solution. With a demand for an accurate solution, we cannot afford to loose this level of performance. We have therefore only evaluated the performance trained and tested on data from the same tissue, and the models cannot be generalized across these different tissue-types .

Chapter 3

Previous Work

There are some studies directly related to our project, both concerning cell type classification and detection of overlapping cells in images. In this section we present a selection of especially relevant studies.

3.0.1 Cell type classification

Classification with kNN- and Bayesian-classification

A study from 2004 [89] used image analysis techniques for segmenting and classifying cell nuclei. They classified the cells into the four categories: mesothelial cells (ME), lymphocytes (LY), granulocytes (GR), and macrophages (MA). They used a set of eight features consisting of: area, perimeter, mean grey value, standard deviation of grey value, circularity, eccentricity, bending energy, and fraction of surrounding cytoplasm. For the first seven features they used images with Feulgen stain (FEU), which is a DNA staining procedure. For the last feature they used May-Grunwald-Giemsa (MGG) staining, which is a morphological staining. For the classification procedure they tried both a Bayesian classifier and a kNN-classifier and used a grid search for parameter optimization. The main problem with this study is the lack of data. All available training data were: 384 ME, 176 LY, 38 GR, and 44 MA nuclei. Because of this they trained and validated on the same set using the leave-one-out procedure. The best accuracy after the grid search was 86.1 % for the Bayesian classifier and 87.5 % for the kNN-classifier. It is problematic that they only report their optimal result, because with no separate test they tend to be optimistic.

In this study they had no class to represent cells to be ignored. They segmented their cells using B-Spline Snakes [10], and split overlapping nuclei using a concavity criterion, where they checked if the line between the deepest concavities had an angle of near 90 ° to the principal axis. Additionally they checked if the area under the line was relatively dark compared to the rest of the cell (see figure 3.1).

3.0.2 Infrared spectroscopy can differentiate tissue types

A different approach used Fourier transform infrared (FTIR) spectroscopic imaging, to classify different tissue types [4]. One of the goals of this study was to segment the epithelium from other cell types. Even if epithelium was the prime

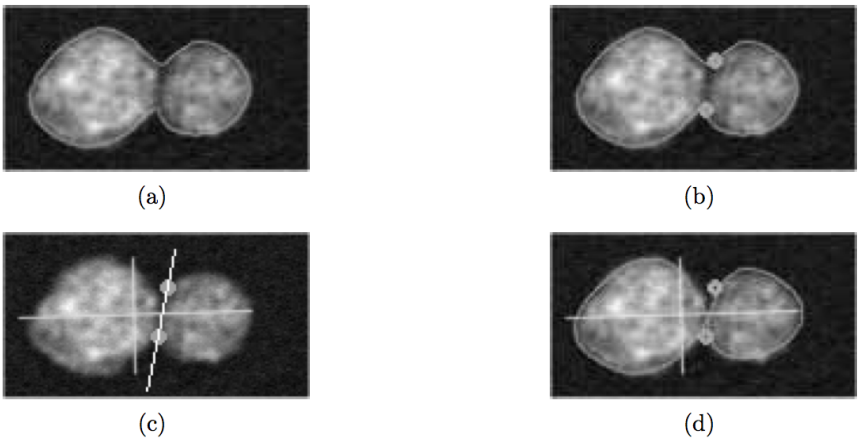


Figure 3.1: (a) contour after segmentation, (b) detected concavities, (c) principal axes and line connecting concavities; the angle between the line and the principal axis is near $90\,^{\circ}$*, the linear path between concavities is relatively dark, (d) split contours, ready for individual segmentation. Copied from [89]*

target, they chose to segment the tissue into 10 different classes: Epithelium, Fibrous Stroma, Mixed Stroma, Smooth Muscle, Stone, Blood, Lymphocytes, Nerve and Ganglion. In addition to taking a spectroscopic image, they also took a digital image in normal visual light of the same tissue. They matched each pixel in the spectroscopic and digital image so they both represented the same exact part of the tissue. Before taking the normal light microscopy image they stained the tissue with hematoxylin and eosin (H&E). With the FTIR spectroscopic imaging technique they measured the absorbance of multiple frequencies of infrared light, for each pixel. For each pixel they ended up with multiple absorbance measures, which they call a spectral profile. They spanned a spectral range of $4000 - 720cm^{-1}$ and had a $2cm^{-1}$ data point interval. In other words, they ended up with 1641 values for each spectral profile, in figure 3.2 one such feature image is illustrated. 1641 values is a huge feature set for further analysis, that will both require much computational power and a large training set. To combat this, they decided to reduce their feature set. They first analysed the pair-wise difference in class distribution, using the area under the distribution curve. They then chose a set 93 features of absorbance ratios, based on this observations.

To further reduce the set of features, they first sorted the features based on low and average pairwise error. The error was calculated as the overlapping area in the feature distribution; calculated with an integral. They then used a sequential forward selection, first selecting the features with the lowest error. With this approach the could examine the effect of adding an additional parameter. For each added feature they applied a classification and evaluated the classification using receiver operating characteristic (ROC) curves. They found that the accuracy of the classification flattened out after 20 features. They then used a "leave-one-out" method on those features and found that the classification

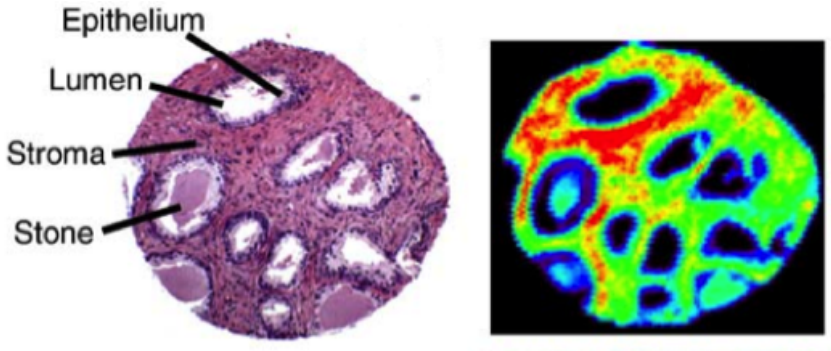


Figure 3.2: ((Right) Visualisation of one feature, from spectroscopic imaging. (Left) The corresponding H&E stained image. Illustration from [5].

accuracy increased when they left out two of the features; so they ended up with a set of 18 features. For classification they used a discriminant function based on Bayesian probability, where the class with the highest probability given a feature vector is assigned to the corresponding pixel. The probability for a feature given a class is calculated from a sample labelled by experts. They mention that the classification is similar to a Gaussian maximum likelihood model, but they do not assume the data distribution to be normal.

This method proved to be a good approach for separating out epithelium with an area under ROC >0.99, on an independent test-set. This is encouraging results, but we have to remember that this approach is somewhat different to ours. This result is based on tissue microarrays, and not monolayers. Additionally they have combined the segmentation and classification process, so they report correctly classified pixels and not whole cell nuclei.

3.0.3 Excluding cells

Christophe Boudry et al. evaluated three different methods to filter out the cut and overlapping cells [6]. They used biopsies taken from three different tumours; two from breast cancer and one from brain cancer. They used a total of 7120 cells for the training and testing procedure.

The first approach is a global one, using only morphological operations on images containing many cells [7]. To remove overlapping cells, they used a combination of watershed transformation, in combination with the ratio between the minimum and maximum radius. To remove small damaged cells, they used top-hat transforms of different size and intensity. For this method, the comparative study showed a sensitivity of 63% and a specificity of 94%.

The second method is referred to as multiparametric analysis (MA), and was originally proposed by Masson et al [58]. Here they calculated 38 features for each segmented nucleus, and used both gray-level images directly and edges enhanced images. They used 11 features for form factor: area, perimeter, compactness, relative perimeter variation, mean symmetry difference, maximum symmetry difference, mean concavity ratio, maximum concavity ratio, extent of concavity, minimum angle between convex contour points, maximum length between convex contour points. 18 intensity factors (gray-level features): min-

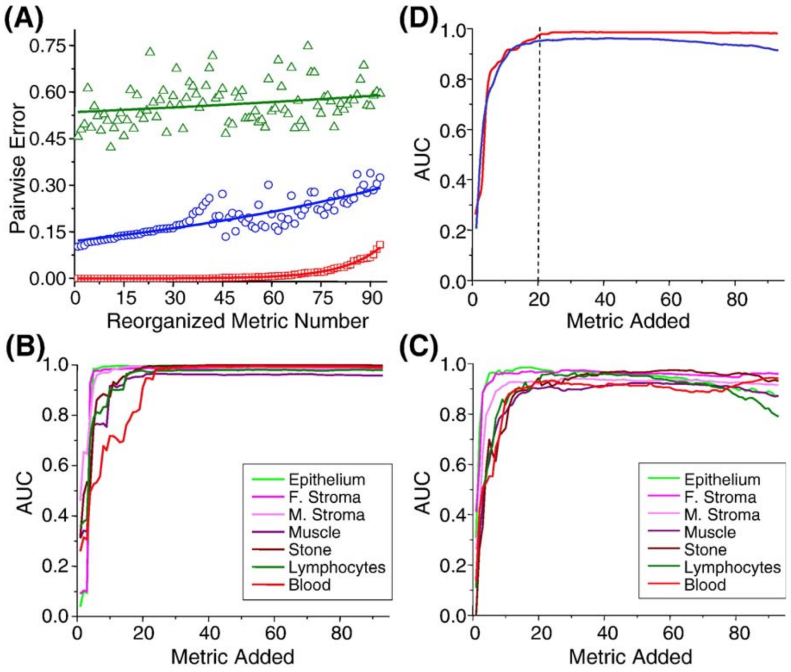


Figure 3.3: (a) The metrics ranked by pairwise error. (b) Evaluation from the forward feature selection process. (c) The same process repeated for an independent test set. (d) Averaged AUC for different tissue types. Illustration from [5].

imum extinction of the object, maximum extinction of the object, extinction range, skewness, Kurtosis of the extinction density within the object, 1st, 2nd, 3rd and 4th moments of the object for original and edge enhanced images. They also included 9 texture features, were 4 features were calculated from isotropic GLCMs (described in section 6.4.1), using the 8-neighbors of the pixels. The 4 features calculated from the GLCM were entropy, energy, contrast and inverse difference moment, described by Haralick [37]. They used five classes of epithelial cells, based on appearance. They also defined five groups of unwanted cells: lymphocytes, plasma cells, granulocyte nuclei, fibroblast cells, unidentified non epithelial nuclei. Finally they defined on of nuclear debris, with overlapping cells and cut nuclei. They used many classes, but they only report the results for separating out epithelial cells to be analyzed. In other words they use a filtering approach similar the current project, but are not interested in detection of cells in K2, K3 or K4.

For each group they calculated the mean and variance in the 38 dimensional space, so they could represent each group by a multidimensional ellipsoid. All groups except the group for overlapping and cut nuclei were represented by such ellipsoid. A sample nuclei is then classified to the closest group or labelled as debris if the distance is larger than two standard deviations from all group means. In the original study they found a CCR of 75% for separating out epithelial cell for analysis and around 70% for the removal of debris. In the comparative study [6], they only measure the ability to separate out debris and found a sensitivity of 75% and a specificity of about 80%.

The last method they investigated were a neural network (NN) with multilayer perceptrons, using the same 38 parameters as the MA approach. They trained one NN for each pair of classes, so each NN could specialize on separation two classes. They used the same classes as for MA, labeling those cells not recognized by any of the NN as damaged or overlapping.

In the comparative study they found a mean sensitivity for the NN approach of 85%, and specificity of about 75%, for separating out the debris.

3.0.4 Summary

The multiparametric analysis described in [58], bear close resemblance to what we want to achieve with our project. They also found quite good results for removing debris, but they only defined overlapping and cut nuclei as debris. In our case overlapping and cut nuclei is only a small part of the total number of nuclei labelled as debris. We also consider their method of "outlier detection" for removing debris to be a dangerous one, as very rare high IOD cells may be more easily removed by this approach. From this study we also get an overview of some features used for cell nuclei classification, although they only provide names of the features and no further description or reference. It may therefore prove hard implement the features and verify that the calculation is indeed the same as in the [58] study. From the comparative study [6], we see that perhaps a method using a range of different features and a flexible classification model such as a neural network can prove favorable.

The study of Würflinger et al. is very relevant to our tissue-type classification. They illustrate that it is possible to do such a classification with just a few features and a very simple classification model. The downsides of their study is both that they use a feature, that is not accessible to us and that their method

of evaluation is questionable. Finally we can see from the study on infrared spectroscopic imaging [4] that there may be other ways of tackling the problem than using visible light microscopy, and that infrared imaging may be a viable alternative.

Chapter 4

Methods

4.1 Fitting an Ellipse

As cells generally have a close to elliptic shape, an ellipse fitting strategy became essential for our project. One of the most commonly used strategies for ellipse fitting was proposed in [22], and relies on least squares minimization. It has proved both robust and effective, and least squares fitting is also a very intuitive principle. The essential principle is to use a restriction on the general conic fitting procedure, so that only elliptic fits can be produced. The general conic can be represented by a polynomial,

$$F(\boldsymbol{a}, \boldsymbol{x}) = \boldsymbol{a} \cdot \boldsymbol{x} = ax^2 + bxy + cy^2 + dx + ey + f = 0. \tag{4.1}$$

To find the least squares solution to this problem we need to find,

$$\underset{\boldsymbol{a}}{\text{minimize}} \quad \sum_{i=1}^{N} F(\boldsymbol{a}, \boldsymbol{x}_i), \tag{4.2}$$

for data points $i \in 1, 2, ..., N$.

To find $\boldsymbol{a}$ representing an ellipse, $\boldsymbol{a}$ needs to fulfill the constraint $b^2 - 4ac < 0$. This is essentially the constraint we are looking for, but this inequality makes the convex minimization problem unnecessary complex. As the scaling of the parameters is irrelevant for our purpose, we can simplify the problem by changing the inequality constraint into the equality constraint $4ac - b^2 = 1$. This can be written in matrix form as $\boldsymbol{a}^T C \boldsymbol{a} = 1$, where C is the matrix:

$$\begin{bmatrix} 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.3}$$

We are then left with the minimization problem:

$$\begin{aligned} \underset{\boldsymbol{a}}{\text{minimize}} \quad & ||D\boldsymbol{a}||^2 \\ \text{subject to} \quad & \boldsymbol{a}^T C \boldsymbol{a} = 1 \end{aligned} \tag{4.4}$$

Where D is the $n \times 6$ matrix $[\boldsymbol{x_1}\ \boldsymbol{x_2}\ ...\ \boldsymbol{x_n}]^T$ With the use of the Lagrange multiplier and setting the differentiated equation to zero, we get the set of equations:

$$\begin{aligned} D^T D\boldsymbol{a} &= \lambda C\boldsymbol{a} \\ \boldsymbol{a}^T C\boldsymbol{a} &= 1 \end{aligned} \tag{4.5}$$

In [22] they prove that this set of equation have exactly one feasible solution. For all sets of points with $N \geq 5$, you will by this approach find exactly one ellipse. We finally used an OpenCV [45] implementation of this algorithm as it proved to be the fastest available.

4.2 Fourier Descriptors

Fourier descriptors [34] can prove highly suitable for describing the shape of the cell contour for two reasons. The descriptors are both easily interpretable and the cell contour is a closed curve, so the assumption of a periodic function hold true.

For our purpose it is essential that a descriptor is both rotation, position and starting point invariant. The position and rotation in a microscopy image is obviously random in terms of the cell label and should not effect the outcome of the labeling process. The starting point is dependent on the contour finding procedure, but this should not effect the shape representation. In it self, the Fourier descriptors are not invariant to any of these measures. This is clear as the Fourier transformation is invertible and no information is therefore lost. The major advantages of this method is that this information is easily identified and extracted.

The discrete Fourier transformation is defined as follows:

$$\hat{X_k} = \frac{1}{N} \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N},\ k \in \mathbb{Z}, \tag{4.6}$$

where X_k is the k numbered Fourier coefficient. From 4.6 it is easy to see that the inverse transformation will be:

$$X_k = \sum_{n=0}^{N-1} x_n e^{i2\pi kn/N},\ k \in \mathbb{Z}, \tag{4.7}$$

4.2.1 Contour Representation

Throughout the literature there have been suggested multiple ways to represent a contour in order to find fourier descriptors. Already in 1961 Fritzsche [31] suggested using the cumulative angular change and a variant of this was purposed in [90]. In [90] they also suggested using the curvature function,

$$\kappa = \frac{|x'y'' - y'x''|}{(x'^2 + y'^2)^{\frac{3}{2}}}. \tag{4.8}$$

They found that the two approaches yielded similar results. A third approach, using complex numbers was first described in [34]. In this study Granlund simply mapped a closed 2D-contour to 1D, with the function,

$$z = x + iy. \tag{4.9}$$

This could be called a "psudo-mapping", as no information is lost, but it creates a possibility of doing a 1D discrete Fourier transform (DCT). As this turned out to be both efficient and have attractive qualities, we found this to be the the best approach for our application.

4.2.2 Interpretation of the Coefficients

Here we elaborate on our interpretations of the Fourier coefficients, based on the descriptions of Granlund [34] and Gonzalez and Woods [32, pp.818–821]. The sections regarding invariance 4.2.3 - 4.2.5, are less formal descriptions of results, also described in [32, pp.818–821] and [34].

The $\hat{X}_0$ coefficient, also called the DC component, has a special position in the DCT. It is simply the sum of the functions values,

$$\hat{X}_0 = \sum_{n=0}^{N-1} x_n e^{-i2\pi 0n/N} = \sum_{n=0}^{N-1} x_n. \tag{4.10}$$

For our purpose this means that we can divide $\hat{X}_0$ by N and find the mass middle of the contour. In other words, the DC component contains the positional information.

The coefficients are complex numbers in similarity with the contour points, but they are not directly related to the points on the contour. A common way to interpret the Fourier coefficient is to use the magnitude spectrum and phase.

Spectrum:

$$|\hat{z}| = \sqrt{\hat{x}^2 + \hat{y}^2}. \tag{4.11}$$

Phase:

$$\varphi = angle(\hat{x}, \hat{y}) = \begin{cases} \arctan(\frac{\hat{y}}{\hat{x}}) & \hat{x} > 0 \\ \arctan(\frac{\hat{y}}{\hat{x}}) + \pi & y \geq 0, \hat{x} > 0 \\ \arctan(\frac{\hat{y}}{\hat{x}}) - \pi & \hat{y} < 0, \hat{x} < 0 \\ \frac{\pi}{2} & \hat{x} = 0, \hat{y} > 0 \\ \frac{-\pi}{2} & \hat{x} = 0, \hat{y} < 0 \\ indeterminate & \hat{x} = 0, \hat{y} = 0. \end{cases} \tag{4.12}$$

Where $\hat{x}$ is the real value and $\hat{y}$ the imaginary value of X_k.

The coefficient for $k = 1, 2, ..., N/2$ could be said to represent circles and further on we will illustrate how. From equation (4.7) or the Fourier basis, we can see that a coefficient $\hat{X}_k$ will give the following parametric contour function in the image domain:

$$\theta_t = \frac{2\pi kt}{N} \tag{4.13}$$

$$\begin{aligned} x(t) &= Re(\hat{X}_k(\cos(\theta_t) + i\sin(\theta_t))) \\ y(t) &= Im(\hat{X}_k(\cos(\theta_t) + i\sin(\theta_t))) \end{aligned} \tag{4.14}$$

for $t \in [0, N)$. If we expand $\hat{X}_k$ to

$$\hat{X_k} = \hat{x} + i\hat{y}, \tag{4.15}$$

we can multiply out the equation (4.14), and isolate the real and imaginary part we get

$$\begin{aligned} x(t) &= \hat{x}cos(\theta_t) - \hat{y}\sin(\theta_t) \\ y(t) &= \hat{x}\sin(\theta_t) + \hat{y}\cos(\theta_t) \end{aligned} \tag{4.16}$$

To find the distance of the point (x, y) to origo, we can simply use the formula $r = \sqrt{x^2 + y^2}$. We insert (4.16) and find the distance,

$$\begin{aligned} r =& \sqrt{(\hat{x}cos(\theta_t) - \hat{y}\sin(\theta_t))^2 + (\hat{x}\sin(\theta_t) + \hat{y}\cos(\theta_t))^2} \\ =& \sqrt{\hat{x}^2\cos^2(\theta_t) + \hat{y}^2\sin^2(\theta_t) + \hat{y}^2\cos^2(\theta_t) + \hat{x}^2\sin^2(\theta_t)} \\ =& \sqrt{\hat{x}^2 + \hat{y}^2}. \end{aligned}$$

This means that the distance from origo is independent of t, and therefore each coefficient has to represent a circle. This also illustrates how the spectrum 4.11 directly relates to the radius of the circle. From 4.14 and 4.13 we can se that the integer k decides how many rounds the curve will move around the circle. With only one coefficient this won't make much of a difference in our image application, as a circle drawn many times over is just a circle, but with multiple coefficients it will give important information about the roughness of a contour. In figure 4.1 we illustrate the effect of two added coefficients, interpreted in a parametric perspective.

Since the sine and cosine functions that make up the Fourier transform are periodic, the coefficients are also periodic. Therefore X_N is the same as X_{-1}, and so on. Each coefficient represents a circle, but combined together the pair of X_k and X_{-k} represent an ellipse. To illustrate this we can use a similar approach as with the circle. The parameterised curve from only the coefficients X_k and X_{-k} will be,

$$(\hat{x}_k + i\hat{y}_k)(\cos(\theta_t) + i\sin(\theta_t)) + (\hat{x}_{-k} + i\hat{y}_{-k})(\cos(\theta_t) - i\sin(\theta_t))$$

using that $\sin(-k) = -\sin(k)$ and that $\cos(-k) = -\cos(k)$. We then get this expression for the radius:

$$\begin{aligned} r^2 =& \hat{x}_k^2 + \hat{x}_{-k}^2 + \hat{y}_k^2 + \hat{y}_{-k}^2 + \\ & 2\hat{x}_k\hat{x}_{-k}cos(2\theta_t) + 2\hat{x}_k\hat{y}_{-k}\sin(2\theta_t) \\ & -2\hat{x}_{-k}\hat{y}_k\sin(2\theta_t) + 2\hat{y}_k\hat{y}_{-k}\cos(2\theta_t) \end{aligned}$$

From this result it is at least possible to see that the distance from origo is $\sqrt{(|\hat{X}_k| + |\hat{X}_{-k}|)^2}$ on one axis and $\sqrt{(|\hat{X}_k| - |\hat{X}_{-k}|)^2}$ on the other.

We have now investigated how the pair of coefficient relate to the width and height of a contour. The last element that we have still not addressed is the effect of the real and complex part of the coefficients. It is easy to see how (4.16) is identical to a 2D-rotation matrix. With this interpretation the real and complex part become the starting point of a circle. The circle is then "created"

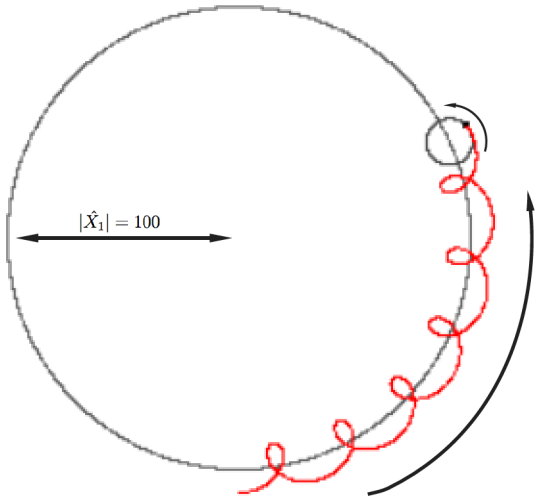


Figure 4.1: The curve drawn by the combination $\hat{X}_1 = 100$ and $\hat{X}_{20} = 10$ is drawn in red; the remaining coefficients is zero. The circle created by $\hat{X}_1$ and $\hat{X}_{20}$ is drawn in gray.

by increasing t, rotating the starting point. So the ratio between the real and imaginary part can be interpreted as the rotation of the circle.

In essence all this means that the Fourier descriptors can be interpreted as a linear combination of arbitrary ellipses.

4.2.3 Position Invariance

As indicated in the previous section, the mass middle point of the contour is represented by the $\hat{X}_0$ coefficient. We can therefore just exclude that coefficient from our analysis to make the descriptors position invariant. That is, we make it position invariant in the sense that exactly the same shape would get the same descriptors regardless of translation. A problem concerning this solution is that the mass middle of the contour may not be the most representative for a shape center. For example if one side of the shape is much smoother than the other, the mass middle of the contour will be closer to the less smooth side compared to the mass middle of the shape. This may result in quite different descriptors for shapes that visually look very similar. In our case this will probably not be a serious concern, as the shapes are quite regular.

4.2.4 Scale Invariance

To achieve scale invariance it would be natural to divide the coefficients by the size of the contour. The most natural way of measuring the size may be to use the average radius. Another measure of size may of course be the area inside

the contour or the perimeter length. If the shape were a perfect circle $\hat{X}_1$ or $\hat{X}_{-1}$ would be exactly the average distance to the center. For our purpose the contours are very close to circles, therefore these coefficients are a good measure of a cell scale. If $\hat{X}_1$ or $\hat{X}_{-1}$ is the correct coefficient depends on whether we store the contour clockwise or counterclockwise.

4.2.5 Rotation Invariance

The Fourier descriptors can be made rotation invariant by only using the spectrum and not the phase of the descriptors, and in that way ignoring the relationship between the real and imaginary component. The downside by ignoring the phase is that you loose the ability to differentiate between many shapes that are visually very different, as so much more information than just the rotation is lost. A contour with high magnitude for high frequency does not necessarily have a jagged edge. It could also have a straight line, and the coefficients is just balanced perfectly. The information regarding how the coefficients are balanced are to some extent lost by ignoring the phase.

In figure 4.2 we have manipulated the phase but kept the spectrum constant. In that way we can investigate what shapes that will be indistinguishable when ignoring the phase. The shapes seem quite similar, but they seem to have a comparable level of jaggedness.

An alternative to using the Fourier Descriptors as features directly, could be to use the Fourier transform to filter out high frequencies and then investigate the differences between the filtered and the original contour. In that way it is possible to keep the information in the phase and make a rotation invariant features at a later stage.

4.2.6 The Effect of Sampling Error

As we are working in the discrete domain, we will get an effect of sampling. At the smallest level the contour is either straight or have a corner of 45° or 90°. This means that we get an extra contribution to the highest frequency, but other frequencies is also affected. One can say that the other frequencies "need" to balance out the effect of the high frequency on the rest of the shape. This means that we at least need to investigate the contribution from the sampling.

The plots in figure 4.3 in the continuos case be zero for all $k \notin [0, 1]$. We can clearly see that that the sampling have an effect and that this effect is not only restricted to the highest frequency. The contributions of the sampling is less for a larger radius, first and formost because the contributions is relatively smaller compared to the radius, but we can also se that the reduction in average magnitude is bigger than increase in radius. This comes as an effect of increased degree of freedom, which means that there are more coefficients to distribute the effect of sampling error to. From this investigation we see that even if we delete the highest frequencies, to get a smoother and more compact description of the contour shape, we may still experience the effect of sampling errors.

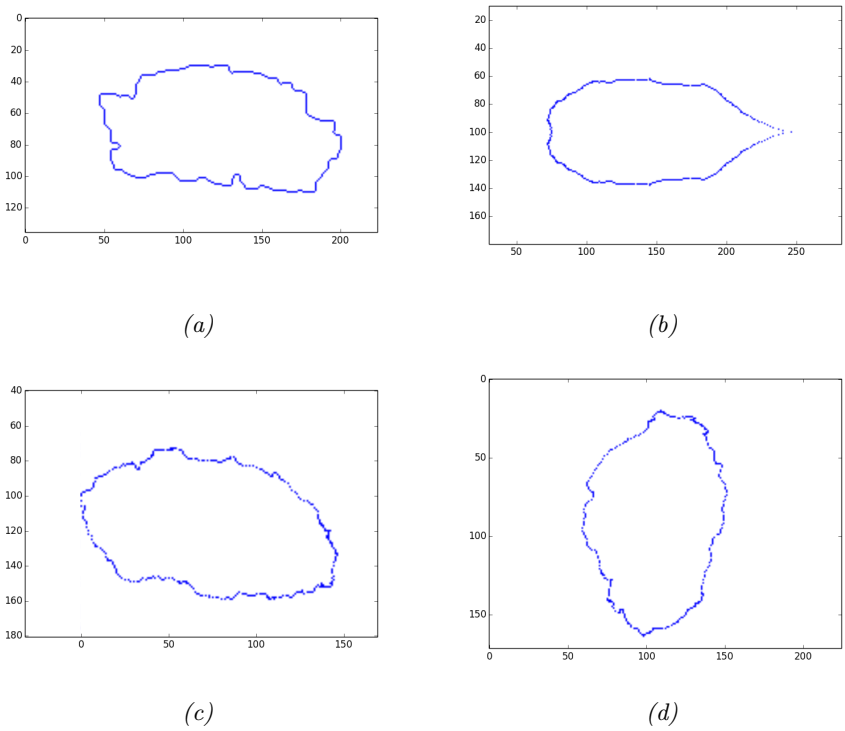


Figure 4.2: Four contours with exactly the same spectrum but different phase. Figure (a) is the original contour of a cell. (b) has the same spectrum but the phase is set to zero. That way it has to be symmetric around the a horizontal axis through the center of mass. (c) and (d) have a randomly generated phase.

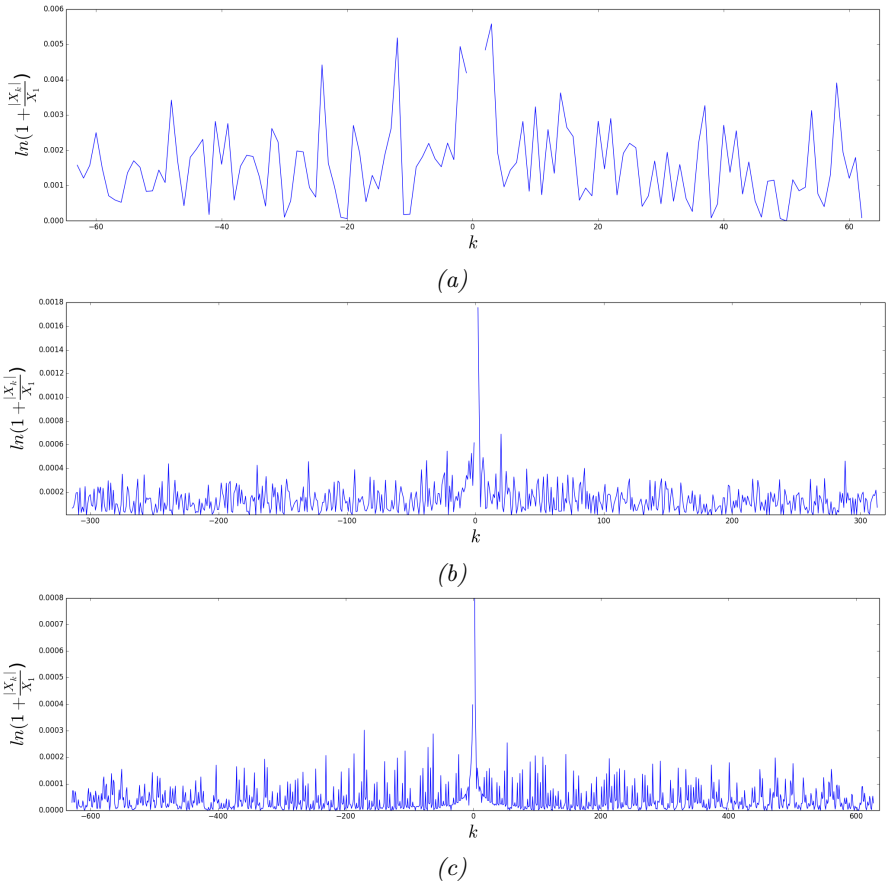


Figure 4.3: The Fourier spectrum of discretized circles of different radius in terms of pixels. We removed the X_0 and X_1 components as they only represent position and radius. Without discretization all other components than X_0 and X_1 should be zero. We divided the spectrum by the radius of the circle and did a logarithmic transform on the y-axis. (a) Is the spectrum of a circle with a radius of 20 pixels, (b) from a circle with radius of 100 pixels and (c) from a circle with a radius of 200 pixels.

Chapter 5

Removing Debris

The main problem of this classification task was the extensive number of cells in K6, with a high degree of overlap against all other classes. There is a large uncertainty about the manual labels, and with a supervised classifier we are totally reliant on the labels in the training set. In other words, we cannot get a better result than the experts. With a complex classification process it is also hard to determine how we got the results, and therefore difficult to trace back the operations leading to an error. This in turn makes it hard to correct the mistakes. If we also use a very general set of features, we often rely on interaction of features to create the separation. This makes it especially hard to understand the relationships between the feature values and the class label. In order to get more control over the process and extract more relevant information about the problem we chose to develop some very specific features, designed only to remove different types of cell debris. We focused on trying to make each feature as independent as possible and making the classification procedure into merely a thresholding.

Rough edges seem to be one of the primary reasons for classifying cells to K6. The problem with these cells is that there appear to be less agreement between the experts on which cells should be removed due to these rough edges. Cut cells are also a very large group of cells and here the experts tend to agree more in their decisions. This means that increased attention on this area may be more fruitful as more debris may be removed this way, and to remove these cell may also be more important. Even though overlapping cells represent a relatively small part of K6, they can inflict major bias. That makes the overlapping cells a primary focus in this chapter.

5.1 Detecting rough edges

There are several difficulties concerning the detection of rough edges. First of all the discretization of the cell contour can make some edges seem rough. Some cells have quite sharp corners that could easily be confused as a rough edge. This is further complicated by the huge variation in cell size. Some irregularities might seem huge for a small cell, but irrelevant for a large cell. With this in mind it is clear that our approach have to take size into account. For very small cells on the other hand the irregularities due to discretization seem comparably

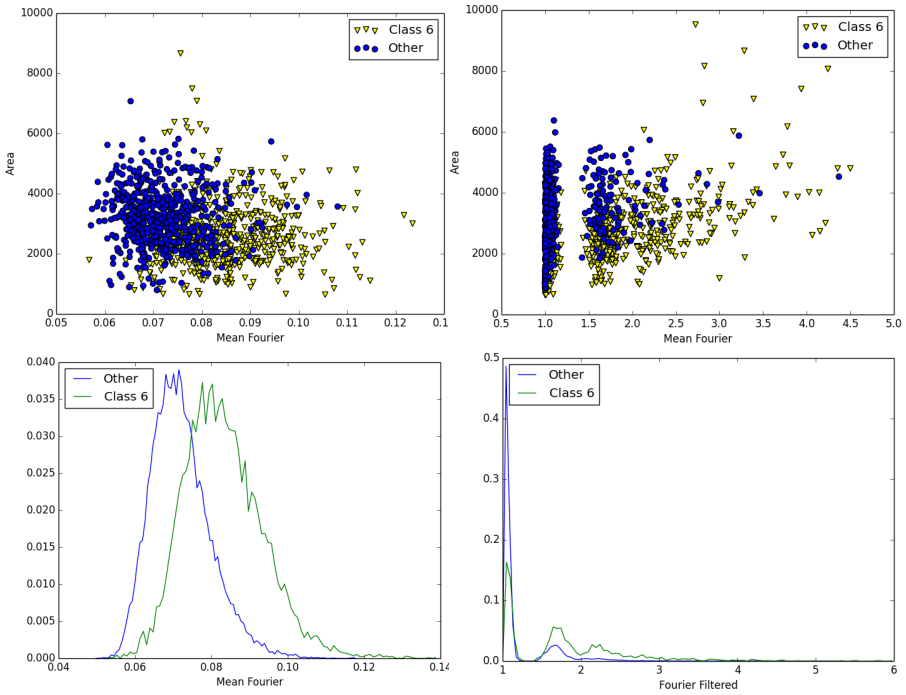
large. Often are detection of rough edges related to detecting cut cells, both because the sharp corners of cut cells appear as rough, and because the cut region often is very rough.

What is obvious is that rough edges are related to high frequencies along the cell contour. Therefore good way to detect these rough edges are to use the Fourier descriptors of a complex mapping described in section 4.2. Then the sum of the spectrum could be used as a simple measure of jaggedness. The different frequencies can also be weighted differently, in order to detect different shapes. We attempted some different weightings, for example a linear weighing between 0 and 1, where the highest frequencies was weighted close to 1. Some attempts of gaussian weighting was also attempted and finally we tried to train a small artificial neural network, to automate the weighing. Unfortunately we found that such a weighting only "overfitted the feature", so only the the training-set was well separated, and as the training procedure was very time consuming we quickly abandoned this approach. The best weighting proved to be to simply remove the 5 lowest frequencies. In other words we sat the coefficients $\hat{X}_i$ for $i = -5, -4, ..., 4, 5$ to zero. Removing $\hat{X}_0$, makes the feature position invariant. Before removing these coefficients, we divided all coefficients by X_{-1}, in order to make the feature scale invariant. The average of the magnitude of these coefficients finally proved to be the most robust feature for detection of rough edges, and we name this feature *mean Fourier* for further use. A downside, with this measure is that we cannot easily find the jagged regions, which might be useful for among other things finding cells with partly damaged cell membrane, or the cut region of a cell.

Another approach using the Fourier descriptors could be to filter the contour, by simply removing the high frequencies. Then we could measure how much the filtered and unfiltered contour deviated, to get an estimate of the jaggedness of a contour. With this approach we also have the opportunity to detect which areas of the cells that are the most jagged. Also for this approach did the mean distance turn out to be the best measure, compared to the summed distance or Hausdorff distance.

From figure 5.1 it is evident that both the distance to the Fourier filtered contour and the mean of the truncated Fourier spectrum can contribute to the discrimination of K6 cells. Still from the histograms it is also clear that the mean of the truncated Fourier spectrum is the strongest feature. We also found that both features match quite well with our visual impression of rough edges, as the non-K6 cells with a high value on this feature also had quite rough edges. Some examples of cells with rough edges can be found in figure 5.2. We found that this feature worked well for P02, PLM13 and PLM14, but when tested on the M51 set, we found that all cells tended to have rough edges, as we can see from the plot in figure 5.3. In figure 5.3, we provide an example of two jagged cells from M51, which both had a high value on our *mean Fourier* feature.

Using the mean of the Fourier spectrum is both simple and quite robust, but it should be possible to find a weighting that correspond even better to the evaluation of the experts. In this study there was not found any such weighting, but with further investigation such a weighing may be found.



(a) *(b)*

Figure 5.1: (a) Is the a scatter plot with mean of the truncated Fourier spectrum on the x-axis and area on the y-axis, and a histogram over just the x-axis. (b) Here we have the same representation but for the distance to the Fourier filtered contour. Both plot are take from a set of 1200 samples of the PLM13 dataset.

Figure 5.2: The leftmost cells was labelled as K1 with a mean Fourier value of 0.11, in the middle there is a K6 cell with a mean Fourier value of 0.105 and the rightmost cell was labelled as K6 and had a mean Fourier value of 0.12.

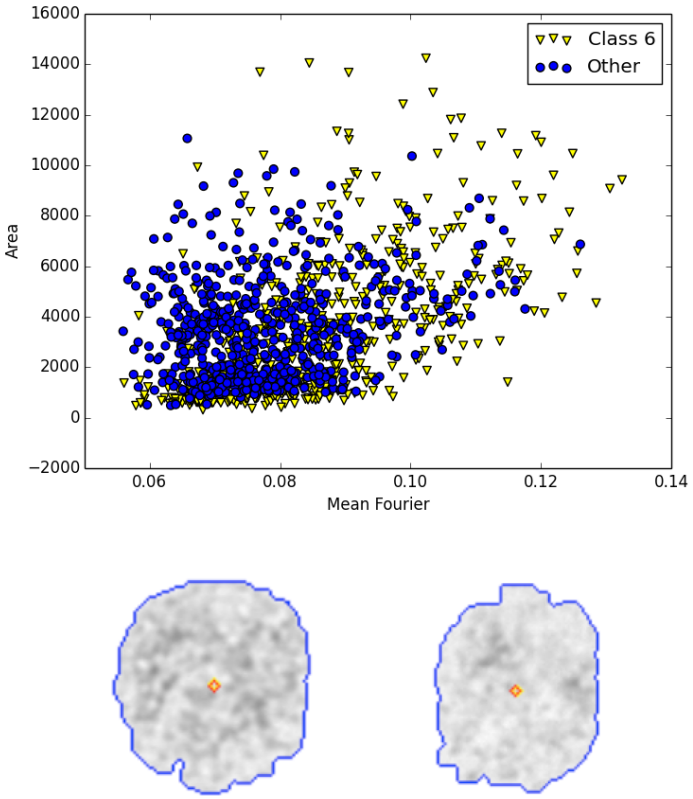


Figure 5.3: Here are two cell with rough edges from M51. The leftmost cell is labelled as K1, while the rightmost is labelled as K6. The plot illustrates the higher overlap of the classes in M51.

5.2 Cut cells

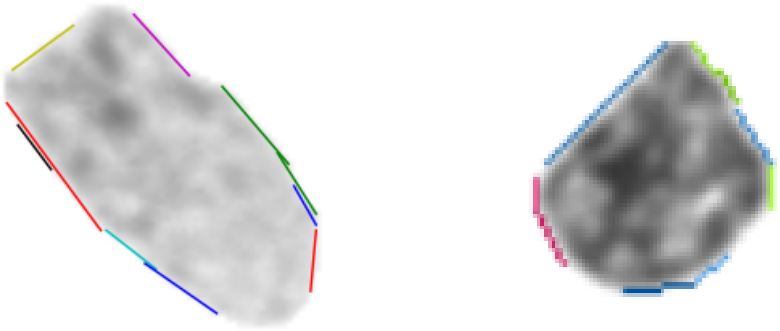
Cut cells are quite common in all the datasets and can also disturb ploidy analysis, as a cut obviously lowers the DNA content for a cell. This can lead cells that originally had high DNA content, to be measured as having normal content of DNA, or it could make histograms that originally should have been classified as polyploid to be classified as aneuploid.

One property concerning the cut cells that immediately comes to mind, is that they have one edge that is closer to a straight line, compared to the elliptic shape of the average cell. Our first attempt to find these cells was therefore to fit a line to the contour of the cell, with the help of a randomized Hough transform. To get an estimation of how well the line fit with the contour, we ran a trace along each line and counted the number of hits. A "hit" is counted when a contour pixel is exactly beneath the line. An example of such fitted lines can be found in figure 5.4. This approach was quickly discarded as stroma cells in K4 and some elongated cells in K1 had much longer straight lines. If we rather scaled by the length of the perimeter, we would only separate very small cells as the chance that a large part of the contour would follow along a line is much greater, this comes from the fact that they are relatively small in comparison to a pixel. The small cells are also often under-segmented in such a way, that they favor straight lines.

In another early attempt we focused on fitting an ellipse to the contour. Either to the whole contour or to the largest part of the contour split by two prominent corners. Then we could measure the distance from our contour to the contour of the fitted ellipse. If we had a strong cut we would then expect the distance to the cut region to be much larger than the distance in general. We found that if we fitted an ellipse to only smaller part of the contour, we would experience some extreme misses on some irregularly shaped cells. Especially triangular shaped cells could get very large ellipses, and the distances would be huge, with or without cuts. If we fitted the ellipse to the whole cell on the other hand we got somewhat less obscure results.

We found that the best approach using fitted ellipses was to combine it with a straight line approach. We first calculated the approximate polygon with the Douglas-Peucker algorithm [20], provided in the OpenCV library, where we allowed a distance of 5 pixels deviation from the cell contour. For each edge in the polygon we calculated the summed distance to the ellipse minus the average distance for all edges. Another approach could of course be to measure the maximum of the minimum distances, called Hausedorff distance [44], but then concavities due to under-segmentation could give a very large distance. The idea is that also the length of lines should count, as we use the sum of the minimum distances, and not the mean or maximum. The approach did work for a small portion of the cut cells, but to find the vast majority of the cut cells we found that we would have to apply other methods. In figure 5.5 we can see how the summed distance to the fitted ellipse, compare to our final measure for cut edges, found in equation (5.4). Most of the cells detected can also be removed by our final measure, but there are also some cells that only the distance to the ellipse can detect. Therefore we include this features in our set for the final supervised classification, but we will not use this for our attempt of thresholding out K6.

Instead of straight lines and ellipses, we focused more on the sharp corner



(a) Cut not detected *(b) Comparably high trace*

Figure 5.4: The colored lines are fitted by Hough transform. The cut in (a) is not the longest line, with the highest trace. (b) Show a small cell with a comparably long straight line.

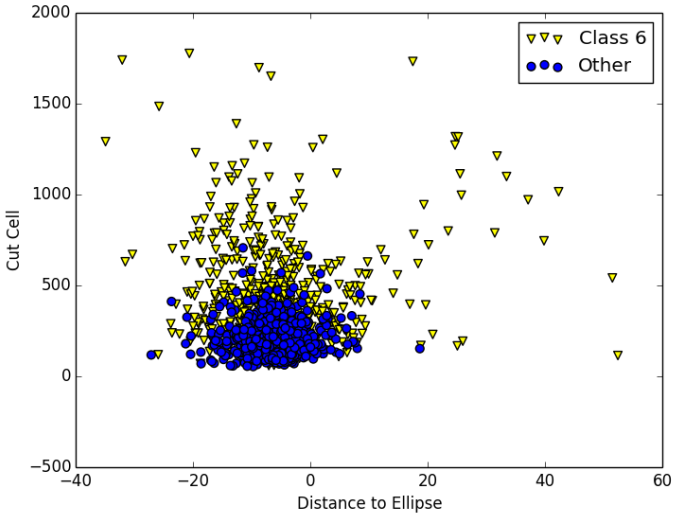


Figure 5.5: We have plotted the distance to the ellipse versus our final measure for cut edges, in equation (5.4). We can clearly see that Cut Cell *is a better measures, and captures many of the cells also captured by* Distance to Ellipse*, but not all.*

of typically cut cells. As a measure of corners in the cell contour we used the curvature, defined as

$$\kappa = \frac{|x'y'' - y'x''|}{(x'^2 + y'^2)^{\frac{3}{2}}}, \tag{5.1}$$

where x and y is the indices for each point along the contour. For later use κ is multiplied with 10^6 to get values that we think is easier to work with and avoid truncation errors. In our application where we use a discretized version of the derivatives, there are some important issues. It is impossible to find a reliable estimate of the derivative by only using the 8-neighborhood, so we needed to apply a smoothing function to the contour. We create a 1D gaussian kernel of size w and standard deviation σ. We convolve it with the Sobel operator once, to create the kernel for the first derivative ΔG, and twice for the second derivative $\Delta^2 G$. The estimates of the first and second derivative, can then be obtained by convolving each of x and y, with the those kernels. We can call the set of point along the contour for p, and we index the contour in a counterclockwise orientation, so p_i is a point on the contour on index i.

We experimented some with the filter size w and the smoothing function, by adjusting σ, to make sure the function was robust to noise from the sampling and segmentation procedures. A good selection of these parameter is essential for detecting the right corners. A low w and σ will give very local corners, and essentially only detect some irregularities along the edge. Too large values on the other hand will not detect the sharp corners we are looking for appropriately, and give highest values to edges that have high rotation, for example the pointy ends of an ellipse. We found that the parameter pair of $w = 41$ and $\sigma = 3$, were best suited to find corners on the scale that we were looking for. Another possible way to detect comers is to use Harris corner detection [38], but this measure proved both more unreliable and harder to adjust.

Now the idea was to use the curvature to detect sharp corner, which again would indicate that a nucleus was cut. For each contour we found the local maxima, as they should indicate a corner. We used the contour from the convex hull of the cell mask, as our equation (5.1) can not separate concave corners from convex corners, and the corners for a cut are obviously convex.

As illustrated in 5.8 this clearly finds the most prominent corners of the contour. As cut cells often have two sharp corners, a simple indicator would be a combination of the κ values from the two most prominent corners. For most cut, both corners are quite sharp, therefore we found that the product of the κ values would be the best approach. This makes cut nuclei stand out from nuclei that only have one very sharp corner due to some artifact, for example slight over-segmentation or just a pointed end. We also applied another method to further increase the difference between actually cut nuclei, and nuclei with slight over-segmentation or pointed ends. This was done by using the relative curvature instead of merely the curvature at some point. By relative curvature we mean that we measured the difference between the local maximum and minimum of the curvature. Our measure for the strength of a corner is then the curvature at the corner point minus the lowest of the closest minima on either side. Since a cut often have a rounded contour on one side and a straight line on the other, we expected the curvature of an appropriate corner to have a very low value on one side. If on the other hand a high curvature point is just a pointed end of a

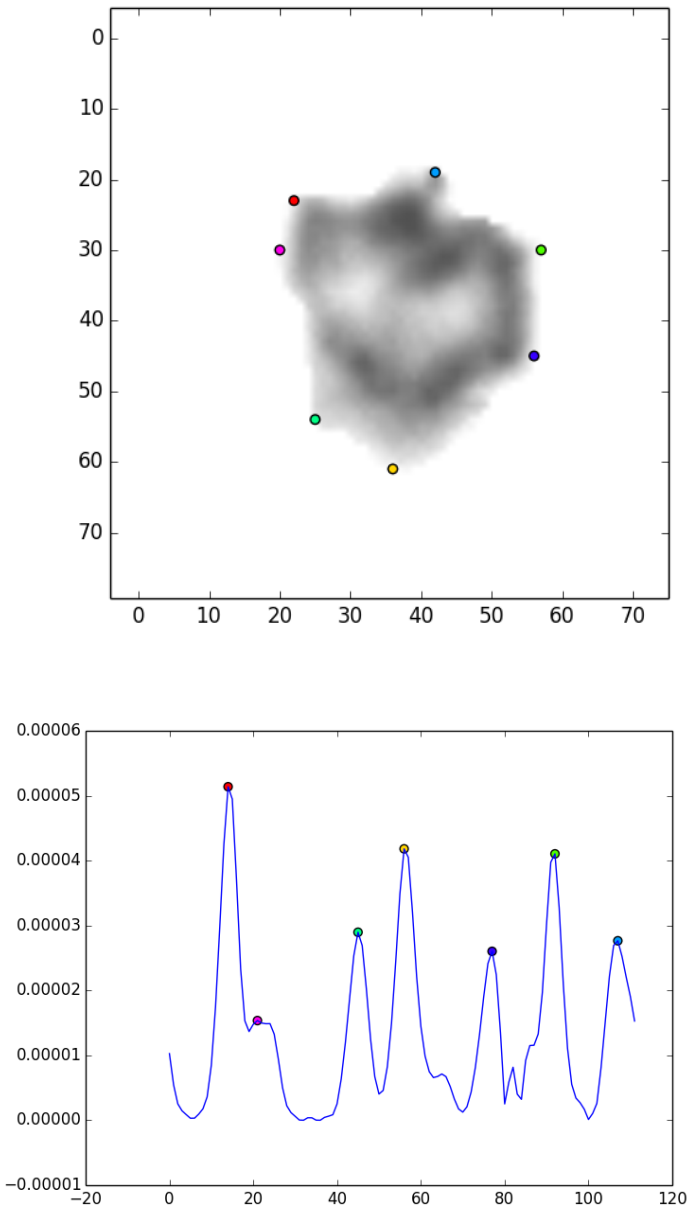


Figure 5.6: Plotting the curvature and the corresponding local maxima. In the top figure the contour points of local maximum curvature is marked with colored circles. In the bottom plot we have the curvature for the contour points, and end each local maximum are marked with circles with colors corresponding with those in the top figure. In other words, circles with the same color are at the same point on the contour.

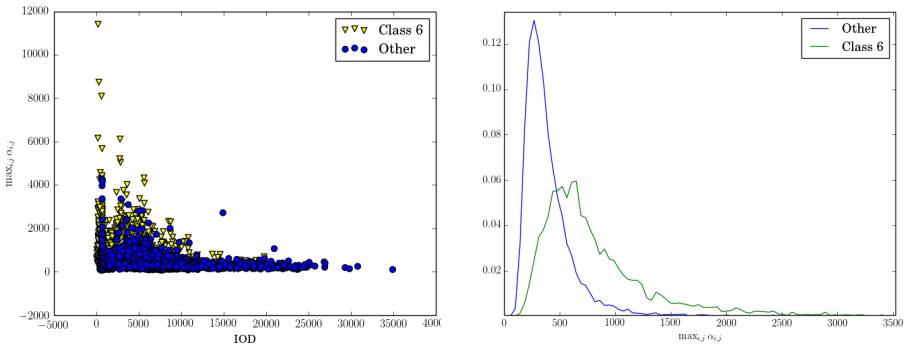


Figure 5.7: Our sum of largest curvatures on the x-axis and IOD on the y-axis. The values on the x-axis is multiplied with 10^6

cell, the curvature on each side is usually also quite high.

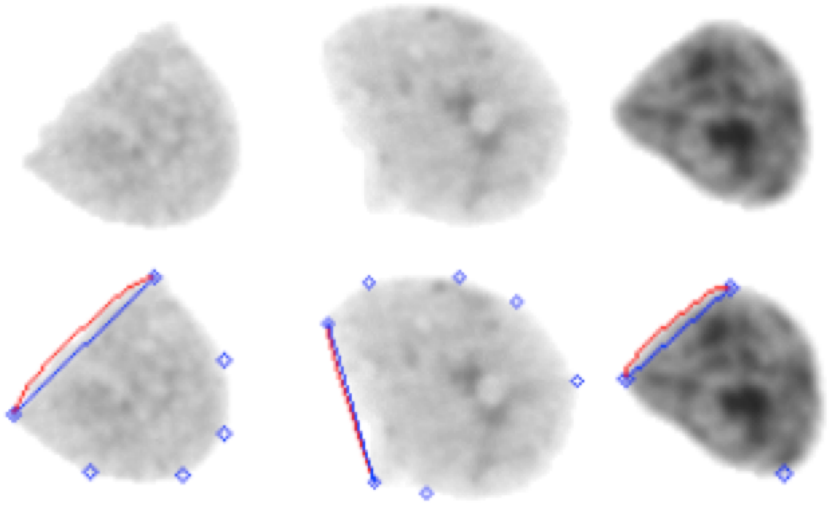
Sometimes the the two points of highest curvature is actually located on the same corner, as the segmentation process sometimes cause corners to have a jagged edges. Small cuts are also usually not filtered out by the experts, so we were not interested in looking for corners in close proximity. We therefore demanded that the corners should be at least $\frac{1}{10}$ of the perimeter apart. We then finally had a measure for the corner strength for a pair of points p_i and p_j, $\alpha_{i,j}$.

From figure 5.7 we can se that our feature give quite good separation, with a threshold of 700 we can remove 37.17 % of the K6 with a loss of 4.9% of the K1 cells. Despite the fairly good separation there are still many undetected cut cells, and many of the detected cells only have rough edges, but are not cut. We also have a problem where some K1 cells have sharper corners in terms of curvature then expected by looking at the cell image. This comes due to over-segmentation, and the fact that corners are especially exposed to this problem. Additionally some K6 cells do not have the "clean cut" we might expect, but look torn. These cells are hard to detect as using the convex hull "blunts out" the corners.

Just the corners is obviously not enough information in many cases, as some cells have spikes and sharp edges without being cut. We want to include the information about straight edges as well, but this information was very noisy, as most cuts have some rough edges similar to the cell in figure 5.8b. We could of course measure the average distance to the contour outside the line, but since we know that the area outside of this line often are brighter and contain less DNA compared to the rest of the cell, we want to incorporate this information as well. To find which side of the line each pixel in the cell mask are, we use the two point of high curvature $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ and the center $\boldsymbol{c}$ calculated as the mass middle point of the cell mask. We then find the two vector $\boldsymbol{v}_1 = \boldsymbol{p}_j - \boldsymbol{p}_i$ and $\boldsymbol{v}_2 = \boldsymbol{c} - \boldsymbol{p}_i$ and calculate for each point $\boldsymbol{p}_k$ in the cell mask

$$M_k = sign(\boldsymbol{v}_1 \times \boldsymbol{v}_2)\boldsymbol{v}_1 \times (\boldsymbol{p}_k - \boldsymbol{p}_i). \tag{5.2}$$

Points counted as *inside* the line have $M_k >= 0$, while points outside the line have $M_k < 0$. We then simply calculated the IOD as described in (6.1), for the



(a) *(b)* *(c)*

Figure 5.8: (a) Is easy to detect, (b) is hard to detect and (c) is confused as being cut. The blue line indicates the selected pair of corners from, all corners found. Corners are indicated by blue circles. The red line show the contour of the convex hull.

area *outside* the line and divided this IOD by the IOD of the whole cell; let us call this fraction for $\beta i, j$. For cells with no area outside the line we set $\beta_{i,j}$ to zero.

Long K4 cells can often have a straight line between each end, that cut through little if anything of the cell and have very high curvature at each end. To avoid detecting these cells as cut, we searched for a feature that could separate them from typically cut cells. We found that the line between the two points of high curvature in K4 cells often passes through quite close to the center of the cell. We therefore calculate the length of the normal, from the line to the center and divide this by the maximum radius of the cell to get the measure $\gamma_{i,j}$. We then have our final measure of cut strength of two contour points

$$C(p_i, p_j) = \alpha_{i,j}(1 - 2\beta_{i,j})\gamma_{i,j} \tag{5.3}$$

β is multiplied by two, as the the maximum amount of IOD that it can cut away is half. The final measure for the detection of a cut cell is then simply the highest of all the values of C for all combination of pairs from the local maximum of curvature

$$Cut\ Cell = \max_{i,j \in I_{max}} C(p_i, p_j) \tag{5.4}$$

where I_{max} is the set set of all possible pair of indexes that corresponds to local maximum in κ.

The plot from this altered technique is illustrated in figure 5.9. With a threshold of 620 we can remove 31.92 % with loosing 5.1 % of the non K6 nuclei. In other words we loose some separability compared to using the α

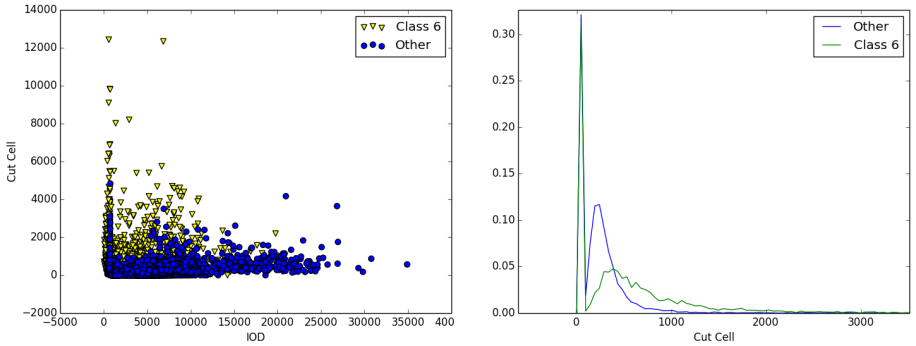


Figure 5.9: In the left plot we have our Cut Cell *measure on the y-axis and IOD on the x-axis. We present a histogram of* Cut Cell *to the right. The leftmost "spike" in the histogram is almost identical for both K6 and other cells, therefore the blue line is partially hidden.*

measure directly, but we detect a larger portion of the cut nuclei and less cell with only jagged edges. Still we cannot detect all cut nuclei. Some triangular K1 cells have quite high curvature on each corner, and a very straight line between them, like the cell in figure 5.10a. They might look very similar to the cut cell. The one thing that seems to make the manual experts label them as K1 is that they have a very clean edge, in contrast to K6 cells that usually have a rougher edge. We did try to use the Fourier filtering approach as described in section 5.1 for detecting rough edges on a single part of the contour. This approach was generally unsuccessful as we found that level of roughness expected to find on a cut edge depended on the general roughness of the cell, and that relationship between roughness of the different contour parts seemed to be complex.

(a) *(b)*

Figure 5.10: Both (a) and (b) are quite similar, and have a "cut value" of around 300. (a) Belong to the K1 group, while (b) belong to K6. One difference is the rougher edge one (b).

5.3 Overlapping Cells

Overlapping cells often have a higher IOD, but we can not use this as feature alone, as we would loose important cells from K1. A prominent feature of

overlapping cells is their concavities, but this proved not nearly discriminative enough. Actually a huge part of the overlapping cells were indistinguishable from the K1 cells, when only considering the image mask. So we definitively had to take patterns inside the cell into account.

Würflinger et al. had already dealt with overlapping cell in [89], described in chapter 3. They first found lines between concavities that was close to 90° on the principal axis. If the linear path between the concavities was relatively dark, they detected a split. We first made an attempt to replicate the approach, but soon discovered that this method would not be sufficient for our data set. First of all, the line between the concavities were often not close to 90° on the principal axis and secondly the edge between the cells did not follow along a straight line, but followed instead a similar curvature as the cell contour.

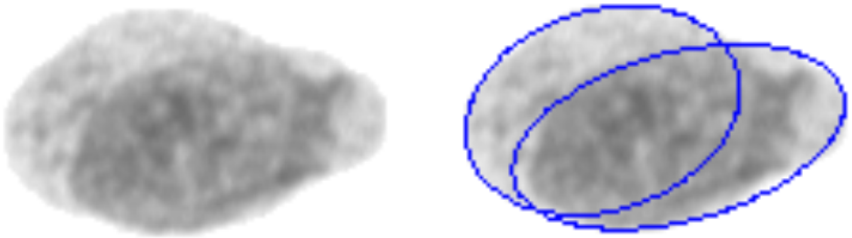


Figure 5.11: An example of typical overlapping cell. To the right we have split the cell contour into two parts, and fit one ellipse to each part. The two points chosen for splitting the contour are both points from a concavity in the cell.

After investigating the manually labelled cells, we found that when it was possible to sense an incoherent edge where cells overlapped, and this corresponded to the expected shape of a cell, then it would probably be thrown into the K6 category. This resembles a subjective contours effect described in [47]. It turned out to be quite hard to estimate the expected shape of the cell contour from a smaller part of the cell, but a very simplified way was to expect the cells to be elliptic. An ellipse could easily be fitted from a part of the contour by the method described in section 4.1 and implemented in the OpenCV library [45]. We would try to find what parts of the contour, possibly belonging to different cells, then we would fit an ellipse to each part of the contour. We found that the best way to do this separation was to use the concavities in the contour. We searched through all all the possible combinations of pairs of points from 10 deepest concavities, along the contour. We would start greedily with the deepest concavities and work our way down the list. For each pair of we could split the contour into two different parts, and then fit an ellipse to each part. To find the area of the cell mask that "belong" to each ellipse, we find the intercept between the ellipse and the cell mask, lets call this new areas A_1 and A_2, and the intercept between the cell mask and both of the ellipses for S. We set two requirements for valid points:

1. Each contour part had to be longer that 20 % of the whole contour.

2. Any two ellipses could not have more than 20 % of their area out side of

the cell mask.

3. The area of S could not be more than 70 % of either A_1 or A_2

If all those demands were fulfilled we would stop the search and proceed with those two ellipses (see figure 5.11).

The idea was to trace the contour of the ellipse, in order to see if it was located in any proximity to an edge of the image. A reliable edge in the image is often not easy to find, as the texture of the cells can be very grainy. This grainy texture can both make the edges incoherent and make "false edges". To remove the grainy pattern we first applied a denoise filtering. A denoise algorithm proposed by Chambolle [12] proved excellent for the job, as it keeps the major structures of the cell quite well (See figure 5.12). Chambolle's algorithm is an approach for solving the total variation minimization problem. The total variation for the discrete image application is defined as the sum of the gradient magnitude for each pixel. The idea is that there exist a *noise free* image u, that we want to restore from a noisy image g. The denoised image is the u that solve

$$\min_{u \in X} \frac{||u-g||^2}{2\lambda} + J(u), \tag{5.5}$$

where X simply is the index range of the image and $J(u)$ is the total variation of u.The denoised cell image, $f_{denoised}$, could then be convolved with a Sobel-operator to obtain the gradient magnitude, and this result was used as our "edge image", f_e.

To avoid finding a path along the contour of the cell we sought to downscale the importance of the edges along the cell boundary. We created a linear scaling of importance, using the distance transform of the cell mask, containing the distance from inside the cell, to the outside. These distances were capped at 7, creating a "trapezoidal weighting function", and then scaled between 1 and 0, to get a weighting image w. The new values of f_e was then set as the point wise multiplication of f_e and w.

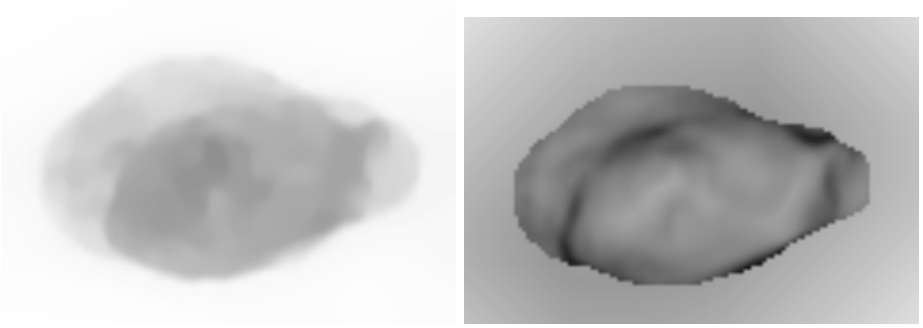


Figure 5.12: To the left we have the cell image after filtering with Chambolle's noise reduction algorithm. To the right we have the final weighted cost image.

As the ellipse rarely matched exactly with the edge of the cells we searched for the best path following an edge in proximity to the ellipse contour, by the use of Dijkstra's shortest path algorithm [19]. We would find the path from one of the concavity-points to the other, where each possible step was to any of the

8-neighbors, not yet found. For input to the shortest path algorithm we created a cost image, weighted by both the distance to the ellipse contour and the edge image. We kept only the part of the ellipse contours where the two ellipses overlapped, and created a mask image where the remaining part of the ellipse contours were set to true and the rest of the image to false. Then we used a distance transform and found and image with the distances to these remaining parts of the ellipse contours. We call this distance image for f_d. The part of f_d that were inside the cell mask, was scaled between 1 and 0, while the area outside was set to 1. The edge image f_e was inverted and then scaled between 1 and 0, lets call this new image $\hat{f}_e$. Then we could create the cost image f_c, as a weighted sum of the edge image and the distance image,

$$f_c = \alpha \hat{f}_e + (1 - \alpha) f_d, \tag{5.6}$$

where $\alpha = 0.4$ was found to give the best result. In figure 5.12 there is provided an example of a cost image. The set of points found by the shortest path algorithm we call P, where each point is represented by a pair of indices (i, j). An example of a detected path can be found in figure 5.13. To evaluate how

Figure 5.13: The path found, marked in red. The path is approximately following the ellipse, but have adapted slightly to the underlying edge.

good the path fits with an edge in the cell, we used the unscaled gradient magnitude, f_e. We simply summed the values of this edge image corresponding to the detected path. This sum was divided by the standard deviation of the gray-levels in the $f_{denoised}$. This is to normalize the result in terms of contrast, as high contrast images will get high values for the gradient magnitude, and any path over the image would get a high value. We found that normalization in terms of standard deviation to be better than using the edge image, scaled between 1 and 0, as this would lead to problems were some black spot in the cell would make the other edge areas negligible. Additionally, if the range is scaled directly, some low contrast images could get a high value, even without a visual edge. A temporary measure for overlapping could then be defined as

$$overlapping_{temp} = \frac{1}{\sigma} \sum_{i,j \in P} f_e(i,j), \tag{5.7}$$

where $f_e(i,j)$ is the value of f_e at the indices (i,j) and σ is the standard deviation of $f_{denoised}$.

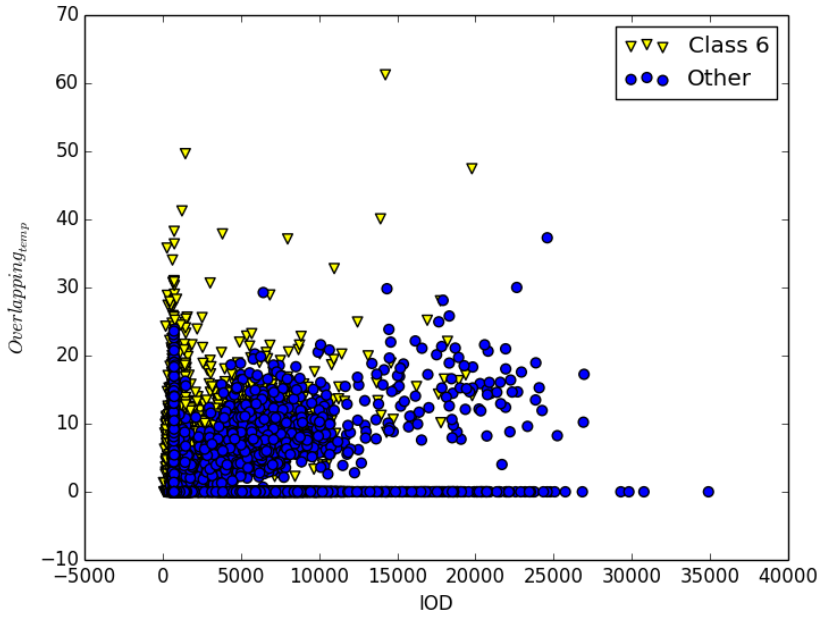


Figure 5.14: Samples from PLM14 scattered with our the $overlapping_{temp}$ feature on the x-axis and IOD on the y-axis. We use no information regarding the concavity of the cell contour.

The plot in figure 5.14 show the effect of $overlapping_{temp}$. Some K6 nuclei is separated, with by $overlapping_{temp}$. In fact we found all cells in figure 5.14 with an $overlapping_{temp}$ value higher than 15 could be interpreted as overlapping. In figure 5.15 there is one K1 cell, with a value of 15.1, that we mean could be labelled as overlapping. We also provide an example of a more clearly overlapping nuclei, and illustrate that the feature roughly correspond to our evaluation of how clearly two cells are overlapping.

With this approach we can capture extreme cases, but there are still quite a few overlapping cells left. One thing we notice is that some cells K1 have a set of very small concavities unfortunately alined with some structure inside the cell, creating a strong edge. To dampen this effect we multiply the value with the depth of the deepest of the concave points that we used to find the ellipse. We can then define our final measure for detection of overlapping cells as

$$overlapping = \frac{d}{\sigma} \sum_{i,j \in P} f_e(i,j), \tag{5.8}$$

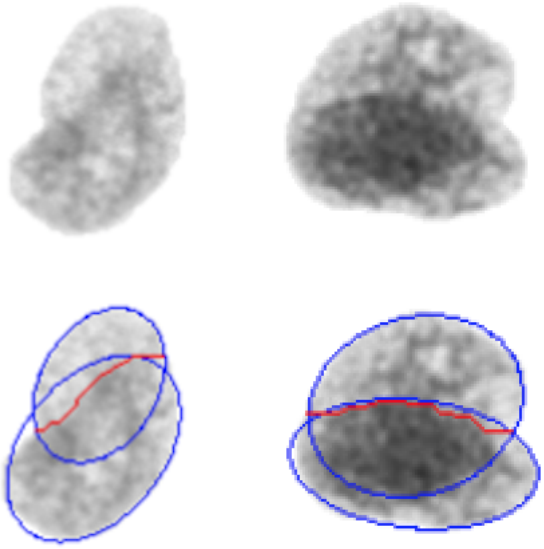


Figure 5.15: Some of the cells with a high value for, the $overlapping_{temp}$ *measure from equation (5.7). The found path is drawn in red, while the fitted ellipses are drawn in blue. The left cell is a K1 cell, but have a value of a value of* 15.1*, while the right cell is a K6 cell with a value of* 21.5*.*

where $f_e(i, j)$ is the value of f_e at the indices (i, j), σ is the standard deviation of $f_{denoised}$ and d is the depth of the deepest concavity in pixels. This gave us some additional separation as we as may be seen in figure 5.16, but some under-segmented cells from K1, with not much of an inside edge, got weighted high because of a deep concavity. Still the final overlapping measure also gave a result closer to our visual interoperation of overlapping cells.

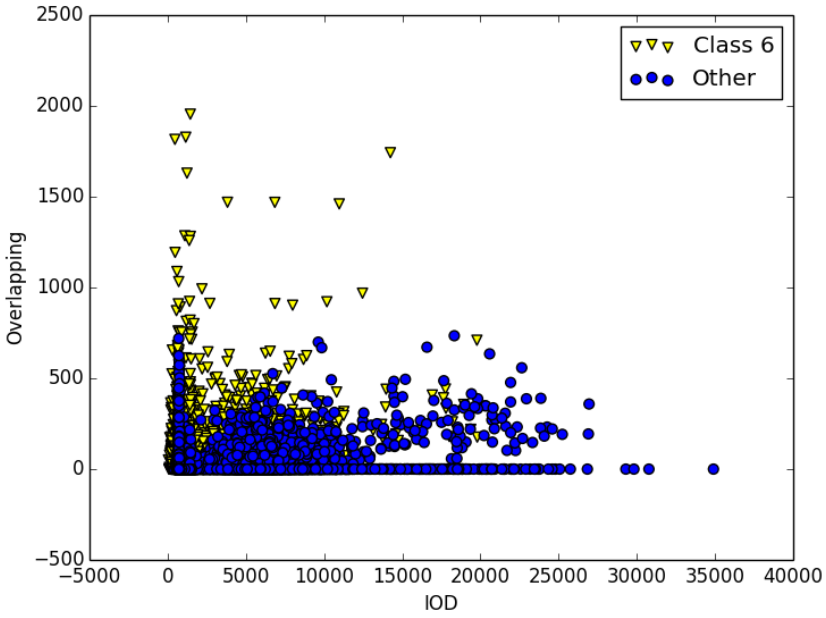


Figure 5.16: Here we plot the same data, from PLM14, as for the plot in figure 5.14, but this time we utilize the concavity information by multiplying the results with the deepest concavity in the contour.

5.4 Over-segmented cells

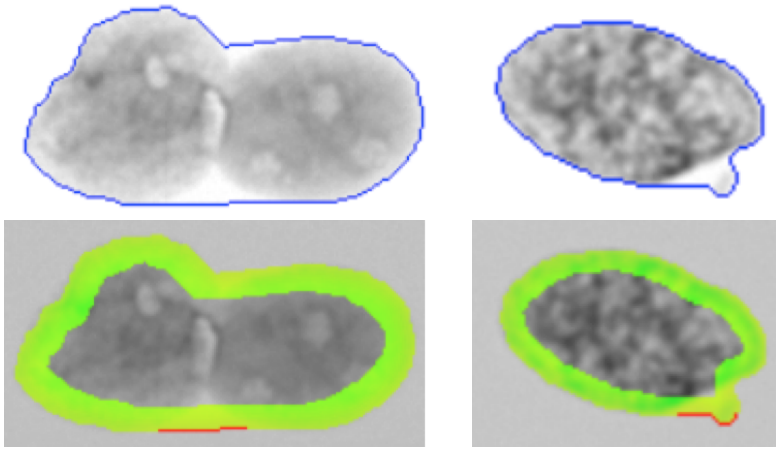
Over-segmented cells could potentially bias texture analysis and other automatic tools for prognostic information. Since many of the overlapping cells not detected in Section 5.3 were over-segmented, an approach to filtering out this category could also help us remove the remaining overlapping cells.

One typical characteristic of over-segmented cells is that they have some regions with very high intensity. This is because the background in the image have a higher intensity than the inside of a cell, and an over-segmented cell contains a region that should have been labelled as background. Our first approach is therefore to look for a region along the edge that has abnormally high intensities. As many images are grainy and have many pixels with max intensity value, we needed to smooth the image to find larger regions.

We first set the pixels outside the image mask to 0 and filtered the cell image with a gaussian kernel, lets call the filtered image B. The qaussian kernel had the size w, where $a = Round(\frac{perimeter}{20})$ and $w = a$ if a is an odd number or $w = a - 1$ if a is an even number. We also filtered the image mask with the same gaussian kernel, where the pixels inside the mask counted as 1, while the pixels outside the mask counted as 0. The filtered mask image we will call M. We could then create a normalized image $C = \frac{B}{M}$. The normalization stopped, concave part where the gaussian kernel had a large area inside the cell from getting a higher value than convex areas.

We then collected the values of C corresponding to the pixels in the cell contour in an array A, where neighboring pixels in the image also were neighbors in the array. We then filtered A with a 1D averaging filter, with length $2w$, to find longer stretches of bright pixels, lets called the filtered version of A for $\hat{A}$. We used the maximum value of $\hat{a}$ as our final measure for a *blurred edge*, or over-segmented cell. In figure 5.17 we have provided two examples of overlapping cells detected by our approach. We have also drawn the area used for our calculation of the blurred edge feature.

The plot in figure 5.18 show that we can separate quite a few cells with this approach, and we found that the most extreme over-segmented cell are separated. Some cells from K2 and K3 are over-segmented but still not filtered out in the manual classification. They can therefore easily be confused with over-segmented cell from K6. We might differentiate theses cell, since they only have a small segmentation error, and this error is often evenly distributed along the edge.



(a) (b)

Figure 5.17: Here we present two over-segmented cells. In the top images we have drawn the cell contour, to illustrate the the over-segmented area. In the corresponding images at the bottom row, we have drawn the area used for calculation of this feature in green/yellow. More yellow colors indicates a relative high value, in the normalized image M. The short segment drawn in red, is the area contributing to the maximum. The average of the pixels under the red segment was used as our blurred edge *feature.*

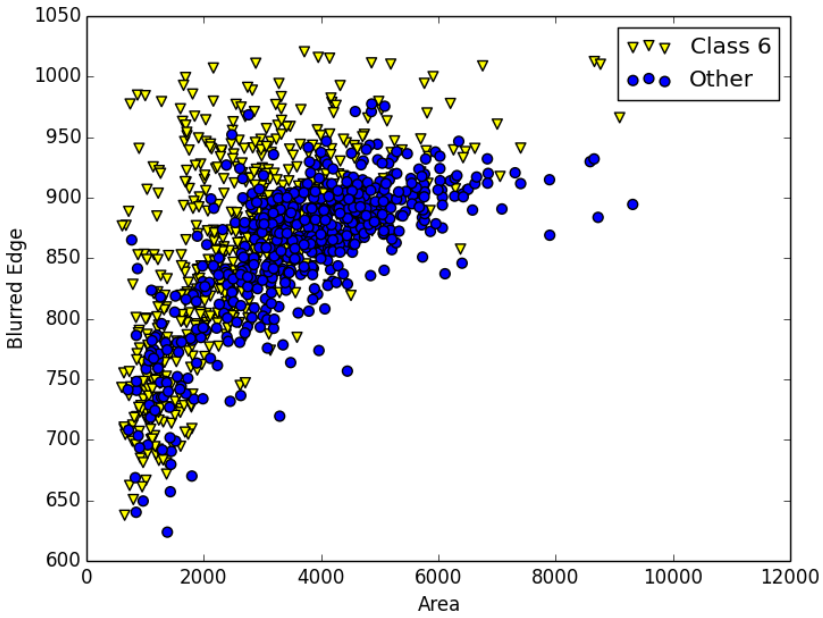


Figure 5.18: A randomly chosen set of 1200 cells from PLM13, evaluated with our over-segmentation measure on the y-axis and Area on the x-axis.

5.5 Blurred Images

As nuclei out of focus were mentioned as on of the reasons for a K6 labeling, we wanted to investigate blur related features. The problem is that evaluation of blur seem to be very subjective. It seems natural that this is a hard boundary to set, as it is not a binary choice, as with overlapping or cut cells. In contrast to those two groups, is the detection of blurred images well represented in the literature, and there is suggested a huge range of features for detecting it [67, 56, 69]. We found it hard to evaluate focus manually, and therefore also hard to intuitively evaluate whether a features was appropriate for our application. Instead we chose to test different focus measures presented in [70] and then simply choose the feature with the highest Mahalanobis distance between K6 and the other classes. We tested the features called: Gaussian derivative, Squared gradient, Helmli's mean method, Variance of laplacian, Steerable filters, Tenengrad variance and Sum of Wavelet coefficients. In table 5.1 we present the Mahalanobis distances calculated from the M51 training-set.

Table 5.1: The evaluation of different focus measures. We measured the Mahalanobis distance between K6 and non K6 samples.

Focus measure	Mahalanobis distance
Gaussian derivative	0.52
Squared gradient	0.30
Helmli's mean method	0.55
Variance of laplacian	0.32
Steerable filters	0.49
Tenengrad variance	0.62
Sum of Wavelet	0.32

As non of the focus measures gave a good separation, and we could not find a clear connection between K6 samples and blurred images, we devote little attention to this topic. As Tenengrad Variance, first suggested in [67], gave the best separation, we kept this measure as a feature. The Tenengrad Variance, called $TENV$ for future reference, was calculated as

$$\begin{aligned} TENV &= \sum_{i,j \in \hat{f}_m} |S(i,j) - \overline{S(i,j)}|, \\ \overline{S(i,j)} &= \frac{1}{|\hat{f}_m|} \sum_{i,j \in \hat{f}_m} S(i,j), \end{aligned} \tag{5.9}$$

where $S(i,j)$ is the gradient magnitude at position (i,j) in an image, calculated with Sobel filter kernels of size 3×3. $\hat{f}_m$ is the set of points (i,j), in the cell mask after it has been eroded with a 3×3 kernel, and $|\hat{f_m}|$ is the cardinality of $\hat{f}_m$. Originally the feature is calculated for a whole image, but our equation (5.9) only use pixels inside the cell mask.

5.6 Notes on the Implementation

The computation of the features was primarily done in Python with the use of Numpy [83]. We used OpenCV [45] for 2D filtering, contour detection, fitting ellipses and morphological operations. For Chambolle denoising we used the implementation provided in the Scikit-image package [84]. For visualizations we used Matplotlib [43]. For the affinity propagation we used an implementation from Scikit-learn [68].

Chapter 6

A Search For Features

In chapter 5 we developed a set of very specific features for removing certain groups of K6 cells. This manual investigative approach gave us good control of the process, but it is both very cumbersome and does not take advantages of modern machine learning techniques that can discover patterns in the data that seem incomprehensible for human observers. For K6 cells it is not likely that there exists such an undiscovered pattern, as the labeling are based on the subjective interpretation of a series of criteria. When sorting the cells in to tissue types on the other hand, there is a definitive biological answer to the classification. This means that there may exist undiscovered patterns in the data.

As our personal understanding of the cell biology is limited, we cannot search for answers through biological understanding. A possible solution is to do a wide search through many different features, designed to capture as many aspects of a cell image as possible. What can be problematic with a wide search is that the use of too many features can cause problems, as the best features gets harder to find. This problem will be later discussed in section 7.1.3. Even without expert knowledge of the field of cell biology, it is still possible to incorporate some a priori knowledge. For example can features presumed to include similar information be excluded and a set of very distinct features chosen. Knowledge from prior studies can be leveraged by reusing features that already have a proven connection to cell biology. In this section, therefor, *no new features are generated*, but we only preform a search thorough a range of know features.

First we present an approach to estimating the DNA content of cell nuclei, as this is a very central feature. The other features we divide into three different categories: Morphological features, features based on first-order gray-level distribution and features related to how the gray-levels are distributed spatially in the nuclei. These three category are discussed in the three sections: *Morphological Features*, *First-Order Gray-Level Statistics* and *Texture and Higher-Order Statistics*.

6.1 Estimating DNA Content

Integrated optical density (IOD) is a measure of DNA content in a cell. It is calculated as

$$IOD = -\sum_{x_f, x_b \in C_m} \log_{10} \frac{x_f}{x_b}, \tag{6.1}$$

where x_f is a pixel value in the cell image and x_b is the corresponding value in the background intensity image.

Such a feature could separate out a huge part of the epithelial cells, as they are primarily the ones with higher ploidy. There are two possible scenarios where including this feature could give problematic results. The first, and most likely scenario, is that all cells with higher than normal ploidy will be classified to K1, as only a very small percentage of cell from other tissue are tetraploid, at any given moment. This situation is not very dramatic, since this involves so few cells the introduced bias would be small. The more problematic situation occurs if diploid K1 cells are less likely to be classified correctly. Then we would be left with a disproportionate amount of K1 cells with high ploidy, and this may affect further analysis. This last more dramatic situation may occur whether we include IOD as a feature or not. The IOD is a very basic combination that depend on intensity and cell area. The information provided by the IOD measure may be added implicitly by other features. It is hard to investigate exactly how a complex classification algorithm combine different features, so an implicit measure of IOD may be used by the classification algorithm, without our knowledge. The important thing to remember is that the classification algorithm makes the decision based on a range of features and it's evaluation is not necessarily biased any more by IOD, than human observers. Perhaps it is easier to include IOD, so we can at least study the effect. Still this is definitively a problem we have to look out for in our classification and analysis.

6.2 Morphological Features

Area

Area is both a simple and relevant feature. It is simply calculated as the number of pixel in the cell mask, C_m,

$$Area = |C_m|. \tag{6.2}$$

One can easily see from figure 6.1 that a huge part of the cells can be classified purely on the basis of the cell area. The combination of *mean intensity* and *area* indicates the ploidy of a cell; shown by curved lines of mainly epithelial cells in the plot. For K4 the high number of *tetrapoid* and some *octaploid cells* for the M51 may indicate that some of the cells labelled as stromal cell actually should be labelled as epithelial. What is clear is that there is no way of separating K1 and K4 just by the area of the cell.

Cell Perimeter

The perimeter length of a cell is obviously related to cell size, but the combination of perimeter and area can still give interesting information. First and foremost it provides information about how circular the cell contour is. If it is also combined with other measures of circularity, it could give information about how jagged our irregular the cell contour is. For example if the cell shape

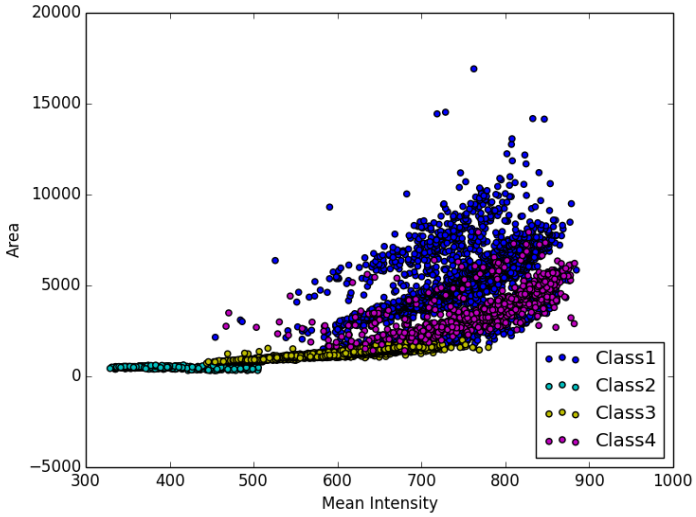


Figure 6.1: Area plotted against mean intensity for a random selected sample from M51. We can clearly see how the cells with different DNA content created patterns related to the cells ploidy.

is quite close to a circle, but the perimeter is disproportionally long, the contour has to be jagged or irregular in some way.

We calculate the perimeter, as proposed by Freeman [25], where we sum the sumer of transitions in the 8-neighborhood contour, but the diagonal transitions are weighted by $\sqrt{2}$, since this is the distance they confer compared to horizontal or vertical transitions.

Circularity

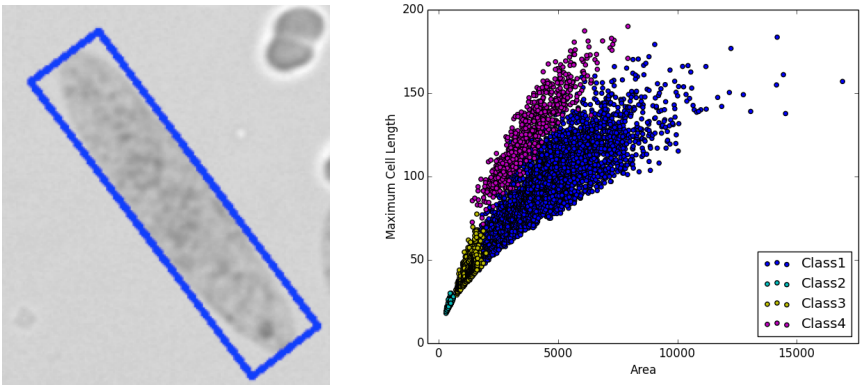
A measure of circularity, combines cell perimeter and area. If we expect the cell contour to be circular we can calculate its radius, either from the cell area or perimeter. When we divide the two radii we would expect to get 1, if the contour were a perfect circle. The expression is created by dividing the radius expected from the *area* (r_a) by the radius expected from the *perimeter* (r_p), and using the squared result

$$Circularity = \frac{r_a^2}{r_p^2} = 4\pi \frac{Area}{Perimiter^2}. \tag{6.3}$$

This means that circularity is close to 1 for cells with nearly circular shape and lower for less circular shapes.

Elongeted cells

There are several methods of measuring to what degree a shape is elongated. Typical measures use the major and minor axis to estimate this. This types of features are typically good for separating out K4 cells, since have a long and thin



(a) *(b)*

Figure 6.2: (a) The minimum area rectangle of a K4 cell. (b) Illustrates how maximum cell length, in combination with area seems to be a good metric for detecting K4 cells.

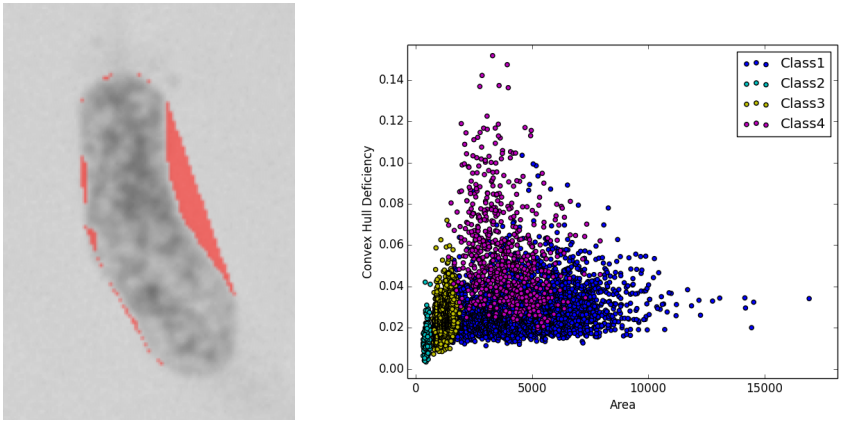
shape. We found that a good alternative could be to include the major axis, or in other words the maximum cell length directly as a feature, and leave it up to the classification model combine this feature with area or cell perimeter. We illustrate this effect in figure 6.2b. The *maximum diameter* feature is measured as the length of the longest edge of the minimum object oriented bonding box and is calculated by using the rotating calipers approach [81]. The shortest edge is somewhat less relevant as it can be quite arbitrary. Some K4 cells can for example be bent, making the shortest edge relatively large. The same problem problem arise for example if we use *eccentricity* as this the combination of the the major and minor axis

$$eccentricity = \sqrt{\frac{a^2 - b^2}{a^2}} \tag{6.4}$$

where a is half the major axis and b is half the minor axis. Still *eccentricity* can be a quite strong feature. Another more complex alternative is to skeletonize [92] the cell mask, find each end of the skeleton and measure the longest distance between two end-points by following the skeleton. This approach is is not affected by bent shapes, but is sensitive to abnormal shapes, for example long thin areas of over-segmentation.

Convex Hull Deficiency

Convex hull deficiency might be most relevant for separating out K6 cells, as they often are uneven and have irregular shapes, and therefore have more concavities. We also expect that K4 cells score high on this feature as bent cells have large convexities. To calculate this feature we used an OpenCV implementation to find the convex hull of a set of points, that is based on the algorithm of Sklansky [77], lets call the set of pixels inside the convex hull C_h, those inside the mask C_m. The convex hull deficiency is then calculated as the cardinality of the intersect,



(a) *(b)*

Figure 6.3: (a) This is a K4 cell, that is slightly bent and has some typical irregularities. The convex hull deficiency is calculated as the area of the pixels marked in red. (b) This plot illustrates how some of the K4 cells are separated out with the convex hull deficiency feature.

$$\textit{Convex Hull Deficiency} = |C_h \cap C_m|. \tag{6.5}$$

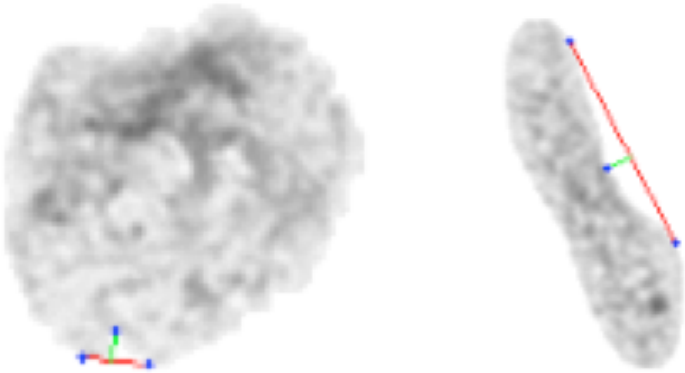
Figure 6.3 illustrates how *convex hull deficiency* clearly contributes to separate some K4 nuclei from K1. In the same figure we also provided an illustration of the convex hull deficiency for a K4 cell.

Depth of Concavity

This feature is strongly related to the *convex hull deficiency* and perhaps left out of the features set. The reason for using this feature is that in combination with the convex hull deficiency, this feature can help us differentiate cell that have a long and shallow concavity, compared to cells that have a short, but deep concavity or several small concavities. K4 cells often tend to have such long and shallow concavities, while both K1 and K6 often have short but deep concavities, typically due to under-segmentation. Figure 6.4 show typical cells in that regard. Figure 6.5, illustrate the difference, where the concavities in the K1 cells are deep, while K4 have large *convex hull deficiency*, but relatively shorter depth of concavity . The depth of a concavity is calculated simply by looping through the contours of the convex hull deficiencies and finding the points on each deficiency that is furthest away from the convex hull.

Jaggedness

We already discussed options for detecting rough edges in section 5.1, in addition to the feature describe in that section, we will test a feature suggested by Maddison [54], and used in the Ploidy Work Station. The feature is a simple way of detecting high frequency changes in the contour. The distance of each pixel in the cell contour to the center of mass is calculated, and represented



(a) *(b)*

Figure 6.4: (a) Is a K1 cell with a short, but deep concavity, while (b) is a K4 cell with a much longer concavity.

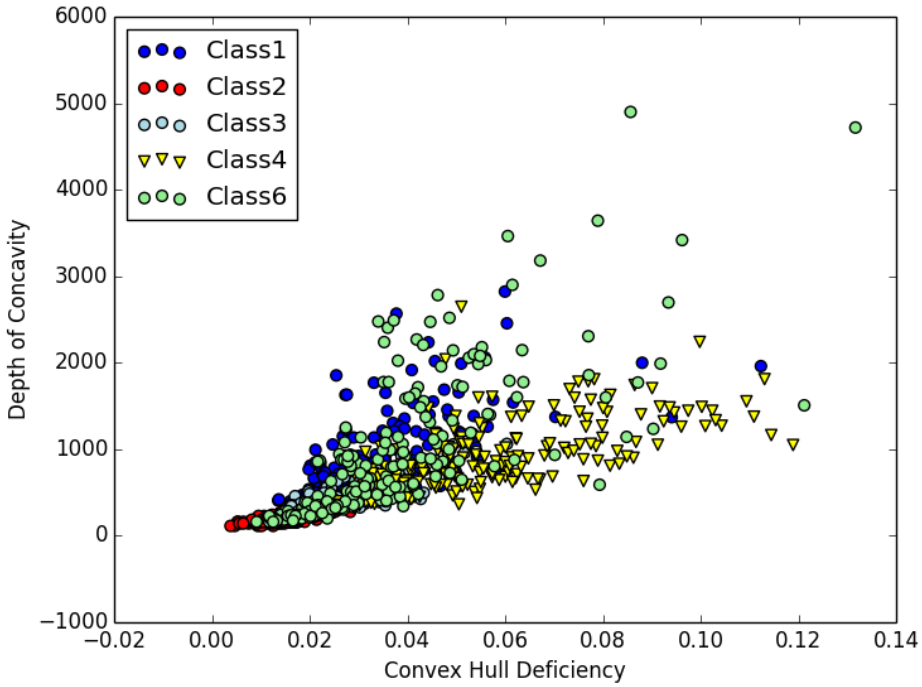
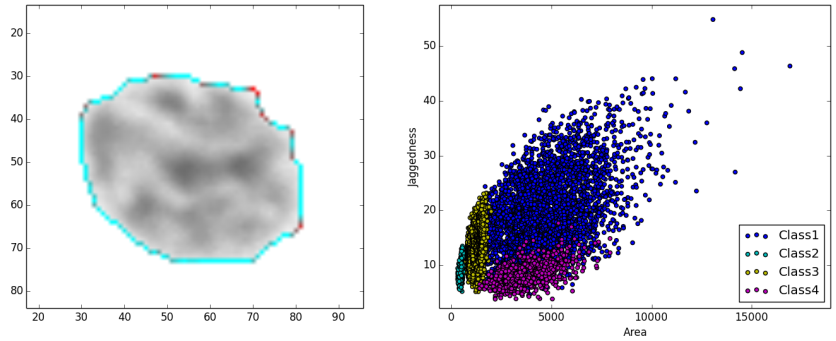


Figure 6.5: 200 cells for each class K1-K4, from the M51 dataset. The depth in terms of pixels can be found by dividing the concavity depth by 256. It is measured on this scale simply to improve performance, as operations with integers are faster.



(a) *(b)*

Figure 6.6: (a) The red area's along the contour have the highest value of jaggedness. (b) Jaggedness seems to be a good way to separate K1 and K4.

by the vector $\boldsymbol{d}$. We then filter d with a median filter of size 5, with a circular boundary function and call the resulting vector for d_m. We then sum the absolute difference between the values in d and the corresponding values in d_m

$$Jaggedness = \sum_{i=1}^{N} |d - d_m|. \tag{6.6}$$

From figure 6.6 we can see that K1 cells tend to have more jagged edges than K2, K3 and K4. This may be because cancer cells are more irregular shaped, but it could also be due to the slightly more grainy texture of K1 cells, that again cause inaccuracies in the segmentation algorithms. Interestingly enough we found that K1 cell also had a higher *Jaggedness* than K6, as illustrated in figure 6.7. The small frequencies detected by this algorithm is not the rough edges that common among K6 cells, but rather segmentation errors in large circular cells or small irregular rites due to slight under-segmentation.

Symmetry

This feature is also a feature used in the Ploidy Work Station and first suggested by [54]. K1 cells tend to be more asymmetrical compared to the other groups. As a simple measure of symmetry we calculate the difference in length between a pair of point, to the center of a cell. We start with the set of points that make up the cell contour $p_i = (x_i, y_i), i = 0, 1, ..., N-1$, where N is the length of the contour. Then we pick two starting points p_1 and $p_{\lfloor N/2 \rfloor}$, as they are approximately mirroring each other through the cell center. If we translate our image such that the average of the contour points is located at the indices (0,0), we calculate our symmetry measure as,

$$Symmetry = \sum_{i=0}^{\frac{N}{2}-1} |||p_i|| - ||p_{i+\lfloor N/2 \rfloor}|||. \tag{6.7}$$

This means that asymmetric cells will have a high value on this measure, while completely symmetric cells will have get a *symmetry* value of zero.

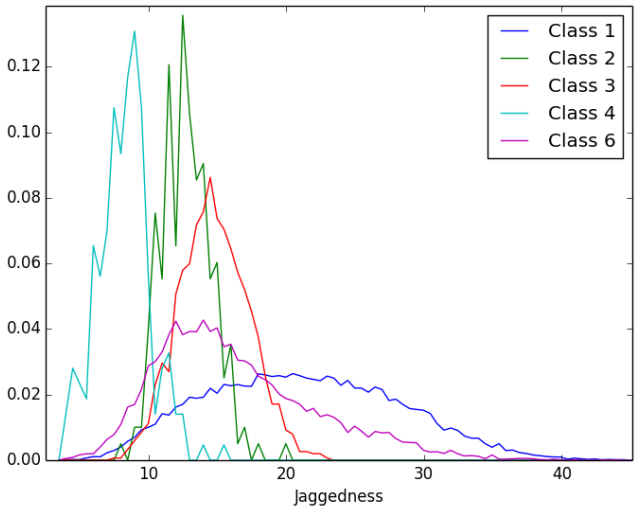


Figure 6.7: The jaggedness, sampled from the PLM13 dataset. K1 have higher jaggedness than K6, as large circular cells will have a larger summed sampling error and that cells with grainy patterns tend to have slight over-segmentation.

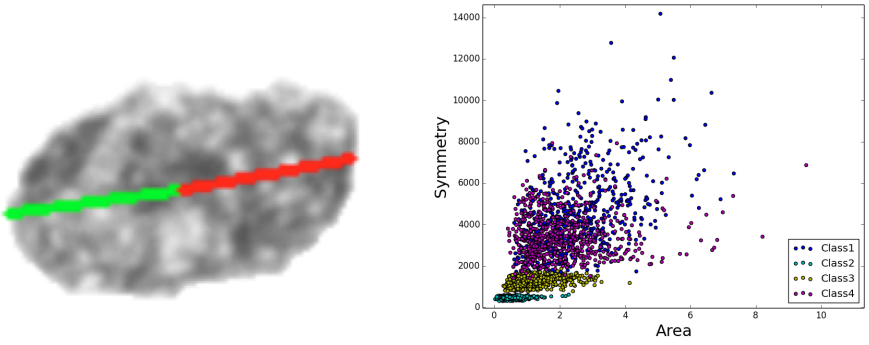
A weakness of this method is that the points will not be exactly opposite; mirrored through the center point. Whether or not the points are symmetric about the cell center, depends on the shape of the cell. However we found that this measure works quite well as an approximate measure for symmetry.

Bending Energy

Bending energy is calculated as the squared sum of the curvature. If p_i is a point along the contour, $i = 0, 1, ..., N-1$, and $\kappa(p_i)$ is the curvature at that point, given by the equation (5.1), then the *bending energy* is

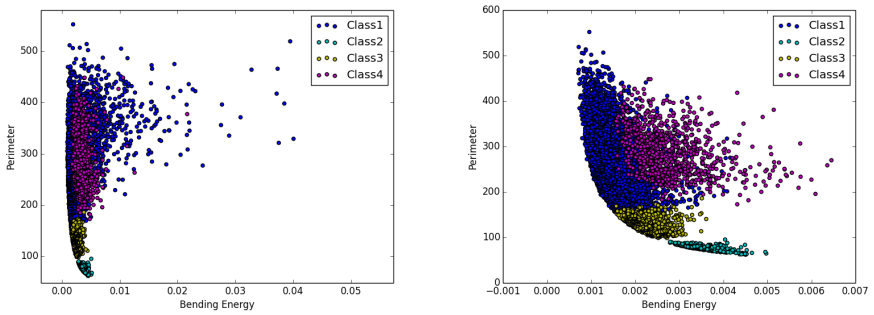
$$Bending\ Energy = \sum_{i=0}^{N-1} \kappa(p_i)^2. \tag{6.8}$$

This can of course be a strong feature for finding *rough edges*, but when separating the classes 1-4, it can actually be a good measure of pointed ends. Since they allow for some rough edges in some of the data sets, especially M51, we found that in order to measure the pointed edges of cells, we needed to use the contour calculated from the convex hull of the cell. Then all concavities are closed up and the effect of pointed ends is increased dramatically.



(a) *(b)*

Figure 6.8: (a) The green and red line are the vectors $(c - p_1)$ and $(c - p_{\lfloor N/2 \rfloor})$. (b) The plot indicates that K1 tends to be less symmetric.



(a) *(b)*

Figure 6.9: (a) Plot of the bending energy of the contour with concavities. (b) Plot of the bending energy of the contour from the convex hull. We can see that using the convex hull drastically improves the separability for this feature.

6.3 First-Order Gray-Level Statistics

First-order statistics are tools that can be used investigate the distribution of gray-levels for an image. They give no information about the position between the various pixels, but only on relationships between the different probabilities for different intensity values in an image. In [59] they found that 4 classes of cell nuclei from mouse liver, had distinct patterns in gray-level histograms. In that study they used transmission electron microscopy and used ultra-thin liver sections instead of monolayers. The cells where categorized into normal, regenerating, nodule and tumor cells. These are of course other categories than those we are operating with, but it is still interesting that tumor nuclei were found to have a distinct pattern in the gray-level distribution, as we hope to find a similar difference in epithelial nuclei versus other nuclei. In those studies they differentiated between the 30% peripheral part of a nuclei and the central 70% of the nuclei. Unfortunately we did not discover a smiliar pattern in our data set. We plotted the histograms for different parts of the nuclei in figure 6.10, and they do indeed look different. However with closer investigation we found that such a division gave no advantages, but rather weakened our predictive power. Instead we decided to focus on the differences in the histograms from the whole cell.

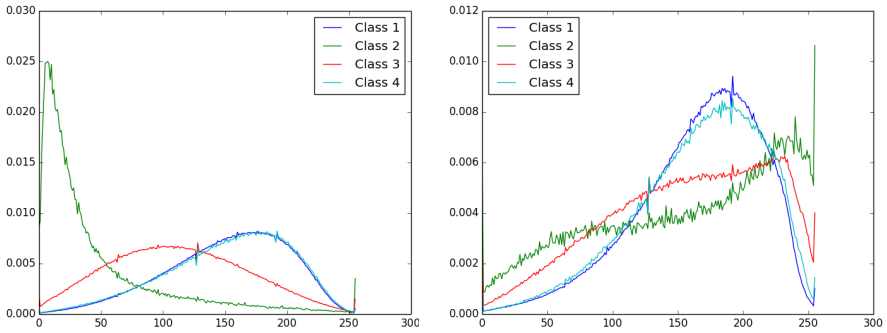
As a quick test to whether these features could be relevant, we created a plot with the average gray-level histogram for the different classes (figure6.11). In the plot the gray-level resolution is reduced from 1024 to 256 colors, in order to get a smoother plot.

Inspecting the Class Histograms

When inspecting the histograms, one have to keep in mind that this does not give us the full information. There may exist subgroups within the different classes, where each of those groups have very different histograms. This also mean that no histogram may necessarily look like the histogram we present in the plot, so assumptions based on the average histograms alone can be futile. For the K6 class we know that there should exist subgroups, as K6 have cells from all the different classes. We can indeed se from the plot that the average histogram for K6 looks like somewhat of a combination of the other plots. In other words, we needed a measure of how much the histograms varied within-class. In order to find such a within-class variation, we first need to find a distance measure for histograms. The Kullback-Leibler divergence [49] is a common way of measuring the difference between probability distributions. We can calculate it as,

$$D(P \mid\mid Q) = \sum_{i=0}^{G-1} \ln\left(\frac{P(i)}{Q(i)}\right) P(i), \tag{6.9}$$

where P and Q are the different normalized gray-level histograms, and G is the number of gray-levels. This measure is not symmetric, the $D(P \mid\mid Q)$ is not necessarily the same as $D(Q \mid\mid P)$, so instead we use the Jensen-Shannon divergence [52] that is based on (6.9), in order to get a symmetric measure. We take the square root of the Jensen-Shannon to get the *metric* [21], we used as



(a) *(b)*

Figure 6.10: (a) Here we have plotted the gray-level histogram for the central 70% of the nuclei. (b) The histograms for the peripheral 30% of the nuclei.

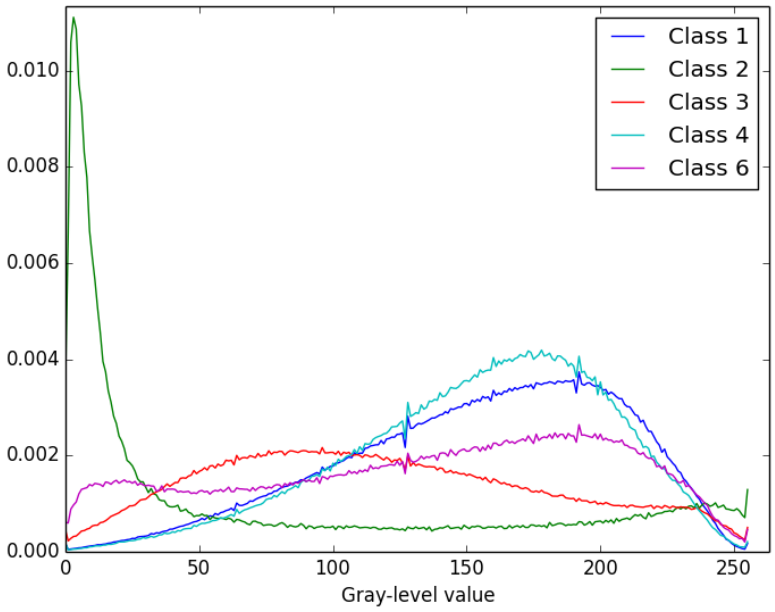


Figure 6.11: The average gray-level histogram for the M51 dataset. We see that K2 mostly consists of very dark pixels, but it also has a small peak with very high intensity value. The high intensity pixels could be there due to the fact that the segmentation algorithm tend to over segment K2 cells. K6 is something in between the other histograms, which is expected as it contains cells from all the other groups.

Table 6.1: The standard deviations from the mean histograms for each class, measured with Jensen-Shannon divergence.

Class	σ
K1	0.22
K2	0.16
K3	0.12
K4	0.27
K6	0.38

our final distance measure D_{PQ}. This metric is calculated as

$$\begin{aligned} D_{PQ} &= \frac{1}{\sqrt{2}}\sqrt{(D(P \mid\mid R) + D(Q \mid\mid R))}, \\ R &= \frac{1}{2}(P + Q). \end{aligned} \tag{6.10}$$

We then say that D_{PQ} is the distance between P and Q. Based on the measure D_{PQ}, we calculated a *standard deviation* σ for each class, which is presented in table 6.1. A table of the Mahalanobis distances between the different classes is presented in Table 6.2.

Table 6.2: The Mahalanobis distances between the average gray-level histograms for the different classes, measured with Kullback-Leibler divergence.

	K1	K2	K3	K4	K6
K1		2.02	1.22	0.16	0.48
K2			2.06	1.84	1.03
K3				1.15	0.50
K4					0.51

From Table 6.1, we can see, as expected, that K6 have a relatively high variation of different histograms, while the cells in K2 and K3 have relatively similar histograms. From table 6.2 we can see K2 cells clearly stands out from the rest of the classes in terms of gray-level histograms, K3 stands out to some degree, while K1 and K4 are not distinguishable, as is also quite evident from figure 6.11.

we can se that the within-class variation for K6 are quite high and similar to the between class distances from and to K6. The distances between K1 and K4 are also very small compared to the within-class variation. This can at least give us an indication that these classes will be hard to separate on the basis of the gray-level histogram. To further investigate if there exists any apparent subpopulations of histograms within the different classes we tested a clustering algorithm on the set of histograms.

Clustering Histograms

Our main purpose of clustering the cell histograms is to find out if the histograms within a class are similar or if there exist groups of very different gray-level

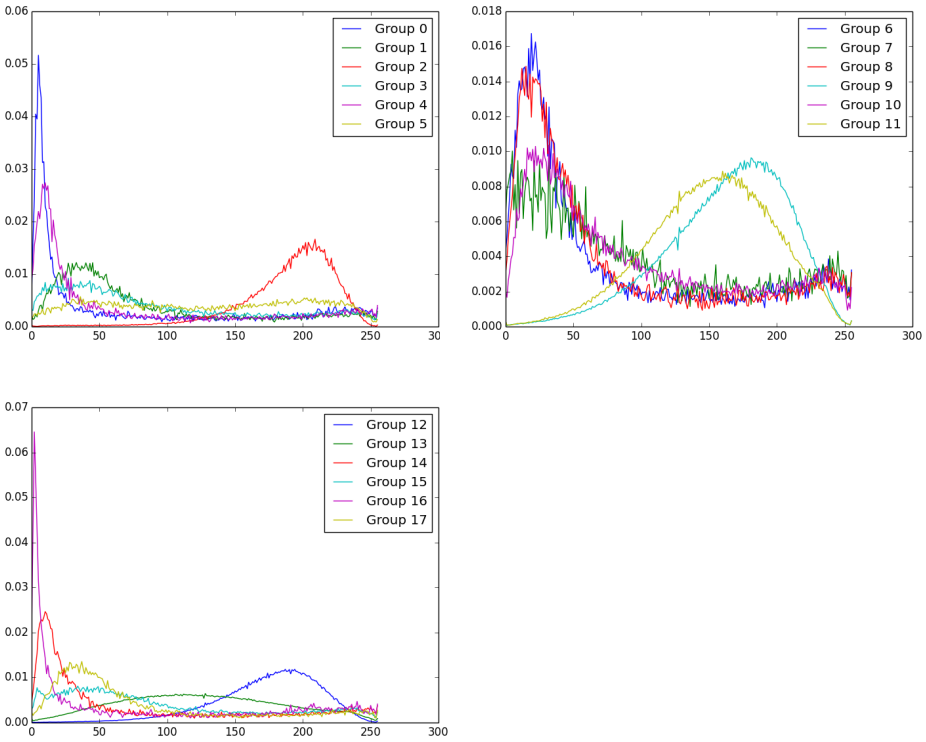


Figure 6.12: The histograms found by affinity propagation.

distribution. If the latter is the case, we will know that the average histogram is not very representative.

We used a clustering approach suggested by Frey et al. in [27, 28], which they called *affinity propagation*. This clustering approach is based on a distance matrix with distances from all samples to all other samples; we still use the Kullback-Leibler divergence as a distance measure. With the clustering of 2000 random samples from the M51 set, the clustering process ended up with 18 groups in total. The average histograms for the different groups are plotted in figure 6.12, while a bar plot over what histograms the samples for each class was labelled to can be find in figure 6.13. We found that to a certain degree the clustered histograms looked similar to the average histograms. We also found that that K2 had almost no overlap in group affiliation with with K1, K2 or K4. For K3 especially, almost all samples belonged to the same histogram group, and this group was one of the larges groups also for K1. K1 and K4 on the had almost a complete overlap, with similar probabilities of belonging to different groups. As Expected K6 had cells in all different groups and the probability distribution for group affiliation corresponded to the relative probability for the different classes.

Our investigation of the gray-level histograms indicate that only K2 can be well separated, by only using the gray-level histogram. The K3 histograms stands out from many of the K1 cells, but still have a quite large degree of

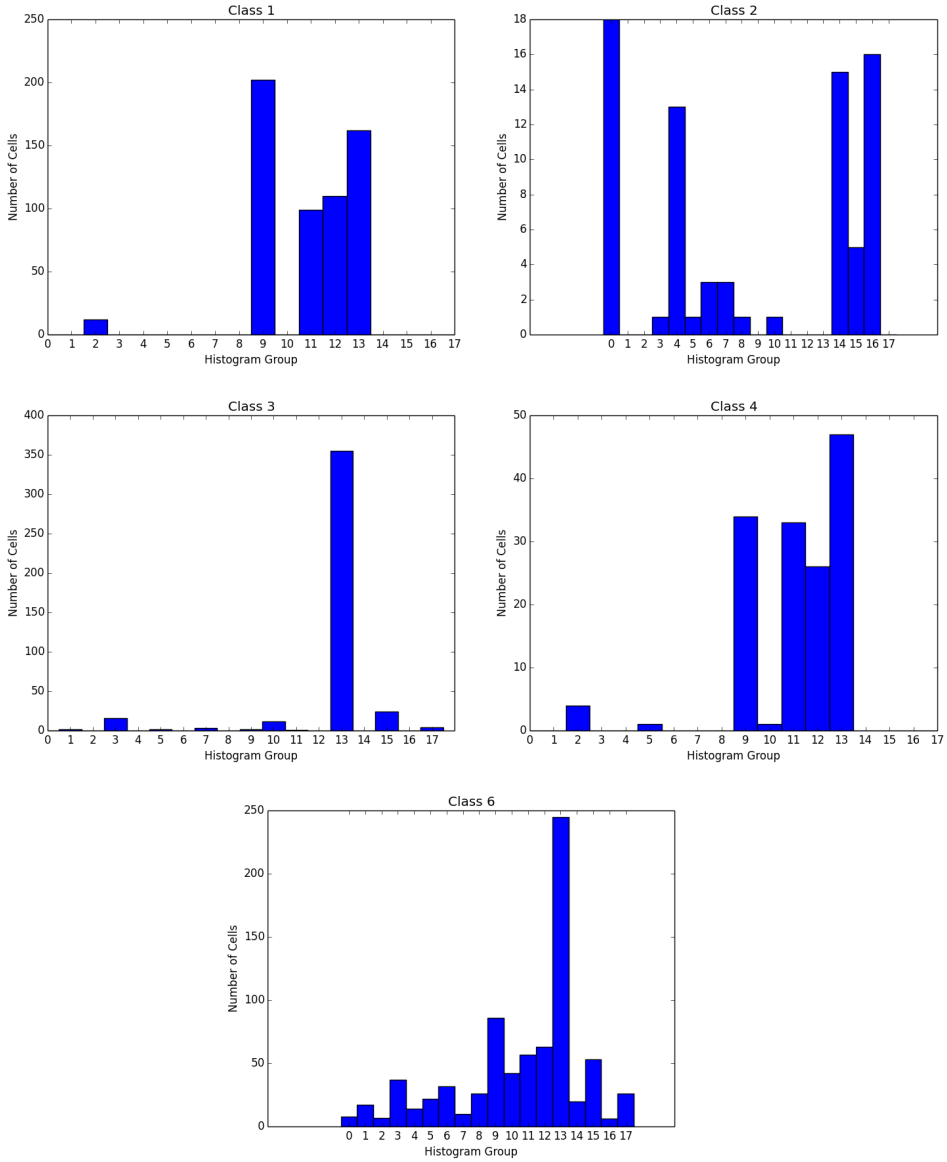


Figure 6.13: The bars represent the number of cells for each class that are clustered to a given histogram in figure 6.12. The class is written on top of the corresponding plot.

overlap. Between K1 and K4 there seems to be no significant difference between the histograms.

Statistical Moments

As K2 and K3 stand out in some ways and also have a comparably small within-class variance they may be the easiest classes to separate on the basis of gray-level histograms. The simplest way to use the histogram shape for classification is to compress the differences in shape into single values. A common way to represent features of such a shape is to use *statistical moments*. In general a moment of degree n around the point c, can be described by the function

$$\mu_n = \int_{-\infty}^{\infty} (x-c)^n f(x) dx, \tag{6.11}$$

For our purpose the *central moments* is most intuitive to use, as they describe the shape, independent of the mean. To get the central moment from (6.11), we simply make sure that c is the mean value of x and that f is the probability density function (pdf) of x. The discrete central moments can then be calculated as

$$\begin{aligned} \mu_n &= \sum_{x=0}^{G-1} (x-\bar{x})^n P(x), \\ \bar{x} &= \sum_{x=0}^{G-1} xP(x), \end{aligned} \tag{6.12}$$

where $P(x)$ is the probability of a gray-level x, and G is the number of gray levels. This means that $\mu_0 = 1$ and $\mu_1 = 0$, which is very logical, but make these two feature useless for classification purposes. Instead we add the *mean intensity*, which is the first moment μ_1 when $c = 0$. For moments of degree three and up, we choose to standardize the moments in terms of standard deviation, making the moments independent of variance. This is done by dividing the n-th moment by σ^n. The standardized moments γ are in other words calculated $\gamma_n = \frac{mu_{n+2}}{\sigma^{n+2}}$. γ_1 is commonly called *skewness*, and provide information on whether the distribution is skewed higher or lower than the mean. γ_2 can be called *kurtosis*, and can be interpreted as describing the "peakedness" of a distribution. Most commonly kurtosis is calculated as $\gamma_2 - 3$, so the kurtosis of the normal distribution becomes zero, but for our classification purpose this is irrelevant.

From figure 6.11 we can see that the *mean intensity* could be a good measure for separating out K2 and K3. We plotted this on the x-axis in figure 6.1 and we can clearly see that it contributes to the separation. A problem is that in our application the mean intensity is very related to the cell area. This is because most cells have the same amount of DNA, for large cells the DNA are "spread thin" and with our staining procedure these cells will look brighter. Most of the information provided by the mean intensity is therefore related to the ploidy of the cell. In other words we have to look out for the same problems as with IOD in section 6.1.

Higher degree moments like *variance*, *skewness* and *kurtosis* could also be relevant for the classification. The pdf of K2 and K3 are relatively wide and should therefore have a higher variance compared to K1 and K4. The K6 average

histogram would also have a quite high variance, but as K6 consist of subgroups we can not say this for certain for the class as a whole. All the histograms are somewhat skewed but K2 most of all, while its clear that K2 are more peaked and therefore will have a higher kurtosis, while K3 will have a relatively low kurtosis. It is harder interpret the effect of even higher degree moments, and therefore hard to determine their descriptive power by investigating histograms. What is clear is that the different moments can indeed separate between the classes to some extent. What is more unclear is whether each moment provide any additional information. Do we need different moments to separate different cells or is it one moment that can give all the separation by it self. To investigate the effect of moments further we used a more brute force solution. We evaluate the moments *mean*, *variance*, *skewness* and *kurtosis*. Using higher order moments will probably contribute little information, and may only add noise to our model, as the histograms vary relatively much within each class.

We tested all possible combination of the 4 features, on a part of the M51 training-set with equal class distribution, with a random forest classifier and cross-validation. Variance alone gave 71% CCR, mean intensity gave 67%, skewness gave 51% and kurtosis gave 41% CCR. Generally a random forest get a better accuracy for added features, but we found that adding kurtosis to the set of the three other features only gave 0.1% increased accuracy, and in some cases even lowered the accuracy. We therefore decided to exclude kurtosis in later classifications.

Entropy

Entropy is another measure we can use to draw information from the gray-level histogram and is said to measure the unpredictability of information or how much information the histogram contain. For our purpose we can say that it measures if there are any dominating gray-levels, that covers much of the cells, or whether all gray-levels have similar probability. We calculate the entropy with the function

$$Entropy = -\sum_{i=0}^{G-1} P(i) \log P(i), \tag{6.13}$$

Where G is the number of gray-levels and $P(i)$ the probability for a gray-level i.

We know that K3 often have many different gray-levels occurring with similar probabilities. This is reflected in the plot in figure 6.11, where we can see that the red distribution is more flat and wide. Some of this information is of course covers by the different moments, but they are aimed at close to normal distributions. The difference is that these moments are influenced by the mean and standard deviation of the plot, while *entropy* does not take the value of the gray-level into account at all, only the probabilities of different gray-levels to occur. In figure 6.14 we can see that K3 have a high average entropy . By visually inspecting the cell images, we would expect the difference to be larger, especially between K2 and K3, as K3 cells have a texture with white spots, where K2 cells are often completely dark. It seems as though a faded edge around the K2 cells, give them a higher range of gray levels than the what is visually apparent.

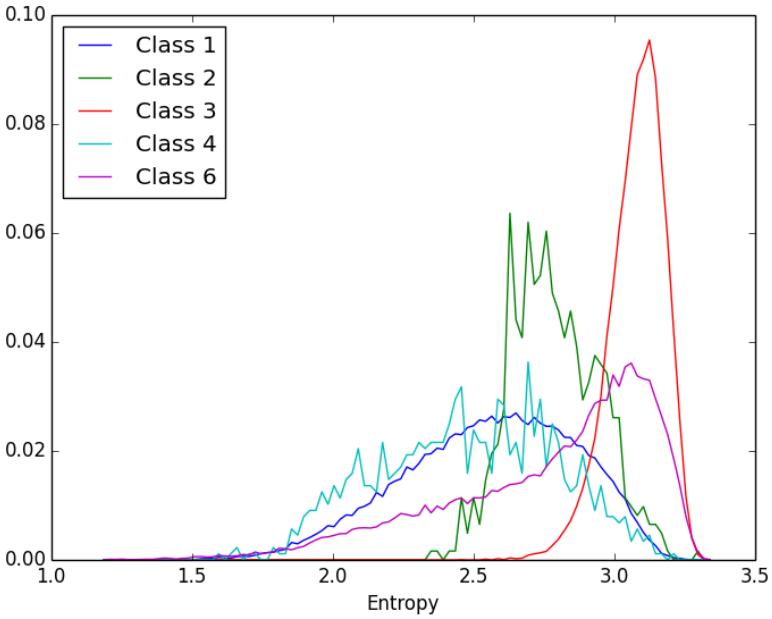


Figure 6.14: A small sample from the PLM14 dataset, where the features are evenly distributed. The cell area is plotted on the x-axis and entropy on the y-axis. We can see that K3 primarily stands out with a high entropy.

6.4 Texture and Higher-Order Statistics

Texture information is a combination of gray-level and spatial information. In other words it describes how the different gray-levels are spatially distributed. Texture analysis of cell nuclei have been shown to provide prognostic information in several studies [1, 2, 3, 63, 87]. We wanted to investigate whether we could capitalize on texture information to improve cell-type classification, but again we have to keep in mind that similar features are used for prognostic information.

There exists a huge range of methods to capture aspects of texture in an image, but we will focus on methods related to *gray-level co-occurence matrices (GLCM)*. We will also investigate whether we can find information related to the more general distribution of DNA in a cell, by using *cartesian geometric moments*.

6.4.1 Gray-Level Co-Occurrence Matrix

A co-occurrence matrix can be used as a tool for calculating second-order gray-level statistics. Second-order means that we investigate the pairwise relationship between pixels, which can give us information about the texture in the image. The matrix has one row and one column for each gray-level in the image. The value at an index i, j, is the probability for a pair of pixels to have intensity value i and j, when the pair have a certain spatial relationship. The spatial relationship is often given by a distance d and an angle θ. If we call the co-occurrence matrix A, the values a each index pair i, j is

$$A_{i,j} = P(i, j \mid d, \theta). \tag{6.14}$$

We get one such matrix for each combination of distance and angle (d, θ). It is common to first reduce the number of gray-levels in an image before producing the GLCM, simply to reduce the need for computational power. In figure 6.15 the process of generating the GLCM is illustrated.

To tap into the relevant texture information we need to find the appropriate distance values, angels and gray-levels. For our purpose it is logical to make the matrices rotation invariant. This is done by simply averaging matrices for different values of θ. We choose the most common approach and use 8 directions $\theta = \{0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°\}$. That is, we only actually calculate half the matrices, as the transition back and forth to a pixel will give the same result. We then calculate the average of the four first matrices and then average that matrix again by its transposed. To decide on the number of gray-levels we both did a visual inspection of the images, to find the how much we could reduce the number of gray-levels without any noticeable difference. Additionally we did a test on a small number of cells, to find out the number of gray levels where we stared to loose predictive power. In both cases we ended up with the same number as many other studies, namely 16 gray-levels. In order to find the best distance values we simply tested all in a reasonable range, $d = 1, 2, ..., 16$.

We only used pixel-pairs where both pixels were inside the cell mask. Without this precaution we could introduce a strong bias, for example for long and thin cells, where many pairs would have one pixel outside the cells, compared to more circular nuclei.

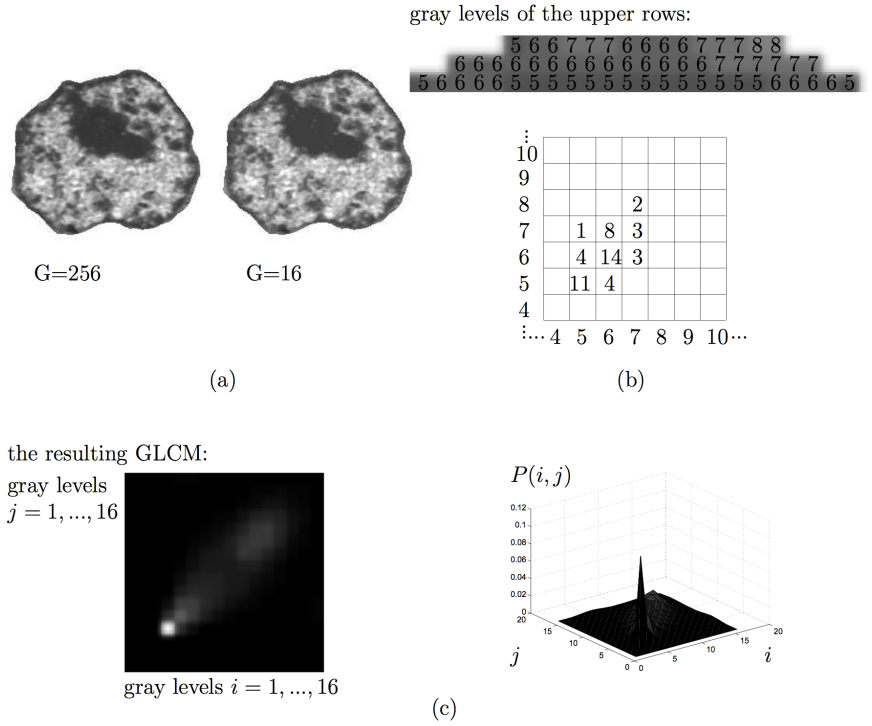


Figure 6.15: Illustrates the process of computing the GLCM for a nucleus. For these figure it is used a inter-pixel distance of 3 ($d = 3$) and an angle of 0° ($\theta = 0$). (a) The cell before and after a reduction in gray-level from 256 to 16 colors. (b) Upper three rows in the cell image and the corresponding GLCM. (c) The final GLCM for the whole nucleus. This figure is copied from [61].

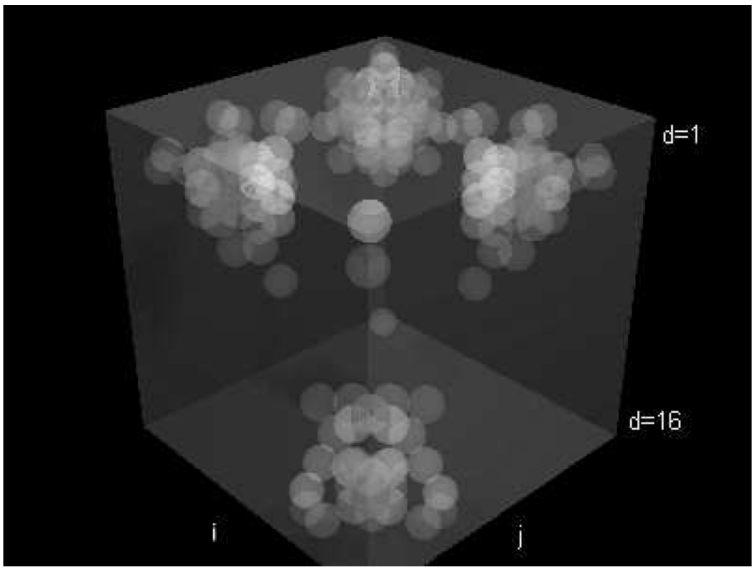


Figure 6.16: The 3D matrix of Bhattacharyya distance of 3D GLCM's of two classes. The blobs are were the Bhattacharyya distance are high. Copied from [86].

GLCM Features

After computing one GLCM for each distance we now have 16 matrices of size 16×16, or 4096 values in total. We could of course use these values as features directly, but that would demand an enormous training set to get a robust model. Such a model would be especially susceptible to changes in gray-level, without an appropriately sized training set. Instead we want to combine these values in a way that gives us a robust and strong separation of classes. If we make sure that values in the same proximate area in the matrix are weighted similar, we can avoid that the classification becomes overly sensitive to small changes in average intensity. Walker et al. [86] found that the values best suited for separating different groups, clustered together both in terms of differences in gray-level, but also in terms of inter-pixel distances. In other words the clusters continued across different values of d. They could in other words create a 3D GLCM and look for clustering differences between classes (see figure 6.16). To measure how well each value were at separating different groups, they suggested to use *Bhattacharyya* or *Mahalanobis* distance measure, for difference between the classes. In a study by Nielsen et al. [65], they used a similar approach for classifying cell nuclei into two different prognostic classes. They separated each nucleus into two parts and calculated GLCM's for each part independently. They also created different groups for different cell sizes, and calculated separate distance matrices for each group. They created features by summing over the GLCM's, weighted by the the squared Mahalanobis class distance for each GLCM value. Mahalanobis distance is in fact a quite common measure for feature selection, so this could be viewed as somewhat of a feature selection or feature weighting procedure. Walker et al. on the other hand used a clustering algorithm, based on the Bhattacharyya distance matrix to find groups of values

with high discrimination that were in close proximity. He then used the sum of these different regions as features. They also proposed to use a genetic algorithm to to find a 3D gaussian weighting of GLCM values, and this actually proved to be the best strategy for find discriminatory features between normal and abnormal cervical cells.

Inspired by these studies we decided to investigate the 6 Mahalanobis distance matrices, generated from 3D GLCMs from K1-K4 (see figure 6.17). By looking at the color bars in the plots, we can again see that the distance between K1 and K4 are much smaller compared to the other pairs. With a max distance of about one σ, there will probably be some overlap, no matter how good features we make. The other plots indicate that we can find a quite good separation, but this has been the case for many other features generated. Additionally it seems that the area of best separation do not change much with inter-pixel distance. Still we have to remember that these distances are calculated from averages and the separation at different values of d may be contributed by different cells.

Predefined features

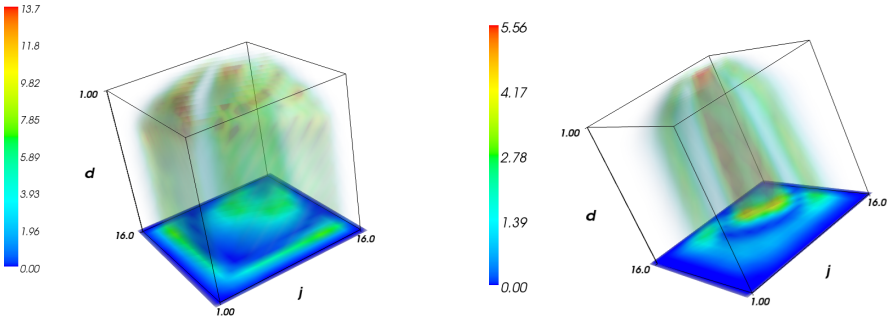
A simple solution to generate a smaller and more robust set of samples from the GLCMs is to use a set of predefined features, known to identify certain aspects of a texture. Haralick et al. and Conners et al. [13, 37] proposed a number of texture features calculated from GLCMs. Some commonly used features are: *contast*, *correlation*, *entropy*, *energy* and *cluster shade*.

$$Contrast = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} P(i,j)(i-j)^2. \tag{6.15}$$

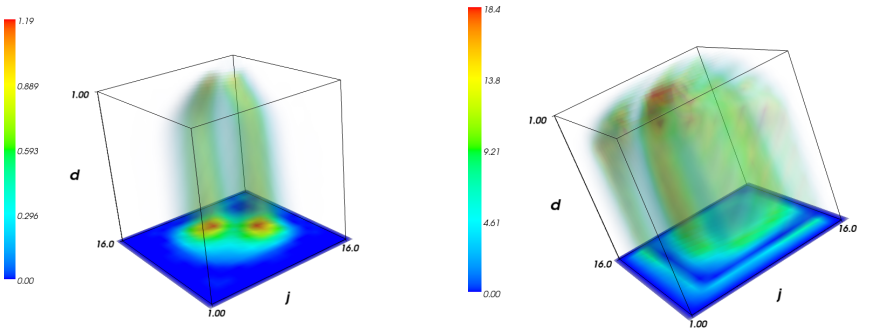
Contrast is basically a measure of how far the the probabilities stretch out from the diagonal in the GLCM matrix. In other words, the contrast will be high if the pixel pairs tend to have very different values. Contrast weight the contribution of values close to the corners $(i,j) = (1,16)$ and $(i,j) = (16,1)$, the highest, and the difference between the matrix for K2 and other classes have an especially high value at these corners.

$$Correlation = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} P(i,j) \frac{(i-\mu_x)(j-\mu_y)}{\sqrt{\sigma_x^2 \sigma_y^2}}. \tag{6.16}$$

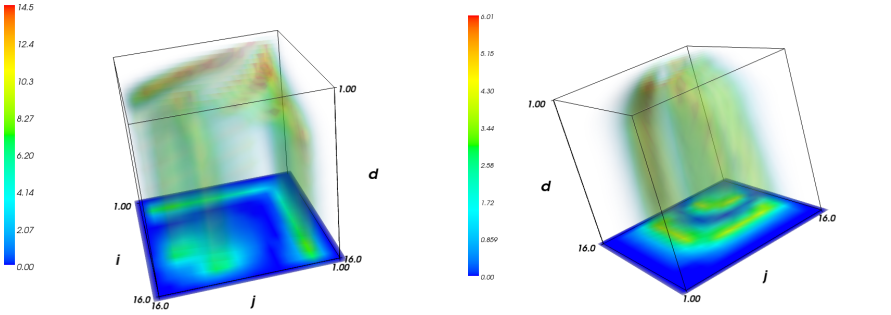
Correlation have the function (6.16), but as we use a symmetric GLCM μ_x and μ_y will be equal as well as σ_x and σ_y. It measures whether i and j move similarly, related to the mean. A matrix with perfect correlation ($correlation = 1$) will only have values on the diagonal, with correlation close to zero the matrix will have values evenly distributed, and a with a negative correlation of -1, only the corners furthest from the diagonal will have values. This is clearly much of the same information provided by the *contrast* features. The main difference is that correlation does not depend on mean value, variance and range, in the same degree as contrast does. With contrast you cannot achieve the maximum value without having both the lowest and highes possible gray-levels, and you need to have a high variance. For correlation you can achieve both perfect positive and negative correlation, even though the range of gray-levels in an image are restricted. When we have already included mean and variance as first-order



(a) $J(K1, K2)$ (b) $J(K1, K3)$



(c) $J(K1, K4)$ (d) $J(K2, K3)$



(e) $J(K2, K4)$ (f) $J(K3, K4)$

Figure 6.17: The Mahalanobis distances J, between each of the average GLCM's. Notice that d range from 1 in the top to 16 in the bottom. (e) is turned so that $i = 16$ and $j = 16$ is facing out, while the other plots have the $i = 1$,$j = 1$ corner facing outwards. The color bar to the left of the plot indicates value and range of the colors for each plot.

statistical features, it makes sense to use features as de-correlated from them as possible.

Energy has the function

$$Energy = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} P(i,j)^2. \tag{6.17}$$

The function (6.17) means that the *energy* value will be high if there are only a few values in the GLCM. That implies that the texture is uniform in some way. It could either be that there are very high probabilities for a few gray-levels in the image or that the transitions are very regular, mostly the same transitions. In our case it will probably be the scenario with a very uneven histogram that are most influential, as a regular pattern are more unlikely with an isotropic GLCM, and especially in a biological setting. In other words it is very related to the first-order entropy.

$$Entropy = -\sum_{i=0}^{G-1} \sum_{j=0}^{G-1} P(i,j) \log P(i,j) \tag{6.18}$$

Entropy is related to unpredictability of a texture. It will have the highest value if the probabilities are evenly distributed in the matrix. This feature is obviously very related to *energy* in the image. It is a weighting function of the image with $weights = -\log P(i,j)$, as opposed to the weighting function $weights = P(i,j)$ for energy. In other words they are quite strongly inverse correlated.

$$ClusterShade = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} P(i,j)(i + j - \mu_x - \mu_y)^3. \tag{6.19}$$

Cluster Shade has a very similar function to the first-order measure *skewness*, especially since $\mu_x = \mu_y$, but it also has an effect related to its second order. First-order skewness has a high absolute value, if it some few extreme gray-level values, far from the mean and only on one side. For cluster shade to have the highest values it needs to have a high skewness, but those extreme values also have to be clustered together, so pixel pairs are picked within the cluster, and both values are extreme. This could for example be used to detect black spots on a bright background or bright spots on a black background.

It seems that *correlation* might be the best GLCM feature for our project, as it is less correlated to our first-order features, but we may want to use several features. We want to avoid using all the features, since they measure similar aspects of the cell, they may only contribute additional noise. In Table 6.3, we present a table with the correct classification rate achieved by each feature alone.

We can again see that for each feature the discriminatory power is not affected much by the distance parameter. None of the features are very good measures, but *contrast* seems to be the most descriptive.

Adaptive Features

To see if it was possible to draw discriminatory power from the GLCMs, we decided to attempt the approach suggested by Nielsen et al. [65], namely using the

Table 6.3: The correct classification rate achieved by each GLCM feature, measured in percent. The test was done on a randomly chosen part of the M51 set, with a random forest classifier.

d	Entropy	Cluster Shade	Contrast	ASM	Correlation
1	52.0	63.1	64.6	68.7	64.4
2	52.1	63.9	64.0	66.8	63.5
3	52.3	65.2	66.3	67.3	66.2
4	**53.3**	**67.2**	68.0	68.6	66.6
5	51.6	67.1	70.8	**69.0**	**67.7**
6	51.1	66.6	69.5	68.8	66.4
7	50.1	67.0	69.2	68.8	65.6
8	51.2	65.9	71.0	68.6	63.3
9	51.1	65.5	70.8	68.1	64.8
10	51.4	65.5	70.3	68.0	64.6
11	51.6	66.1	72.1	66.6	63.3
12	50.9	65.2	71.5	67.3	63.4
13	50.4	64.9	**72.5**	67.3	61.7
14	49.5	65.0	72.2	66.4	61.0
15	48.4	64.0	71.9	65.2	59.6
16	46.9	63.9	71.8	65.0	59.4
Average	50.9	65.4	69.8	67.5	63.8

Mahalanobis distance matrix as a feature weighting. We follow the exact same procedure, except for deciding the cell, based on central and peripheral regions. First we calculated the mean GLCMs for each class $K = K1, K2, K3, K4$,

$$\overline{P}(i,j \mid K) = \frac{1}{N_K} \sum_{n=1}^{N_K} P_n(i,i \mid K), \tag{6.20}$$

where N_K is the number of cells in K and P_n is the normalized GLCM for a single cell. Then we computed the difference matrix

$$\Delta(i,j \mid A,B) = \overline{P}(i,j \mid A) - \overline{P}(i,j \mid B) \tag{6.21}$$

where A and B are two classes, for all combinations of classes K1-K4. We used the Δ matrix to divide each GLCM into two regions, depending on whether the corresponding value in the Δ matrix were positive or negative. The division was done to make sure that positive and negative differences did not cancel each other out. Finally we could compute our distance matrix

$$J(i,j \mid A,B) = 2\frac{(\overline{P}(i,j \mid A) - \overline{P}(i,j \mid B))^2}{\sigma_P^2(i,j \mid A) + \sigma_P^2(i,j \mid B)}. \tag{6.22}$$

For all the 6 possible combination of classes, we then calculated two features, as the sum over each of the regions, weighted by J^2

$$\begin{aligned} F_+ &= \sum_{i,j:\Delta(i,j|A,B)>0} P_n(i,j) J(i,j \mid A,B)^2 \\ F_- &= \sum_{i,j:\Delta(i,j|A,B)<0} P_n(i,j) J(i,j \mid A,B)^2. \end{aligned} \tag{6.23}$$

With the total of 12 adaptive features we tested the discriminatory power of both the features individually and the combined set. The results from Table 6.4,

Table 6.4: A table of the CCRs from a random forest classification of the adaptively generated features. The lower left part of the matrix is the F_- features, while the upper right part is the F_+ features. We found the adaptive features by training on a random selected part of the M51, and these samples were not included in this test.

		High value			
		K1	K2	K3	K4
Low	K1		60.9	56.3	49.2
	K2	83.6		80.5	83.6
	K3	77.4	68.0		75.8
	K4	77.4	65.6	47.6	

illustrate that our adaptive features definitively provide better discriminatory power compared to classical predefined features. The adaptive features will be named after what distance they they are optimized for, and the first class will be the one with the highest values. So adaptive GLCM (1 vs 2) means that we summed the Mahalanobis distance between K1 and K2, where K1 had a higher value than K2.

6.4.2 Cartesian Geometric Moments

It may be that different cell-types differ in how the DNA, or gray-levels in the image, are generally distributed in the cell. Some cells may have most DNA along the outer edges of the nuclei, while others have more of the DNA lumped together in the middle of the cell. It may also be that some cells have the weight of DNA away from the center of the cell, in other words a skewed distribution of DNA. When measuring the distribution of DNA, much of the information is obviously influenced by the cell shape. Since no DNA is outside the cell, the cell shape defines a very distinct boundary for the distribution.

To specify some features, we treat the gray-level distribution in a cell as a 2D probability distribution P(x,y). Then we can use statistical methods to find parameters for the shape of this distribution. As statistical measures of the shape we can use moments, as we did in the section on *First-Order Gray-Level Statistics* 6.3. We can define the Discrete Geometrical Moment as

$$m_{p,q} = \sum_{x=0}^{M-1} \sum_{y=0} x^p y^q f(x,y), \tag{6.24}$$

where $f(x,y)$ is the gray-level value at index (x,y). We say that the moment have order $(p+q)$. The problem with these measures is that they a not translation, rotation or scale invariant. They are basically a description of the distribution along the x and y axis and a combination of the two. In order to get the moments translation invariant, we simply use the central moments

$$\mu_{p,q} = \sum_{x=0}^{M-1} \sum_{y=0} (x-\bar{x})^p (y-\bar{y})^q f(x,y), \tag{6.25}$$

where

$$\bar{x} = \frac{m_{1,0}}{m_{0,0}}, \bar{y} = \frac{m_{0,1}}{m_{0,0}}. \tag{6.26}$$

A set of scale and rotational invariant moments were introduced by Hu [42]. He uses a combination of normalized central moments, defined as

$$\eta_{p,q} = \frac{\mu_{p,q}}{\mu_{0,0}^{\gamma}}, \tag{6.27}$$

with

$$\gamma = \frac{p+q}{2} + 1. \tag{6.28}$$

Since the central moments depend on the size of an image and the gray-level value, it is normalized by dividing by the sum over the whole image ($\mu_{0,0}$). As $\mu_{p,q}$ is multiplied to a power depending on p and q, we also need to multiply $\mu_{0,0}$ to a corresponding power, in order to get a truly scale invariant measure. Hu proposed 7 rotation invariant moments based on the normalized central moments, $\eta_{p,q}$.

$$\begin{aligned}
\phi_1 =& \eta_{2,0} + \eta_{0,2} \\
\phi_2 =& (\eta_{2,0} - \eta_{0,2})^2 + 4\eta_{1,1}^2 \\
\phi_3 =& (\eta_{3,0} - 3\eta_{1,2})^2 + (3\eta_{2,1} - \eta_{0,3})^2 \\
\phi_4 =& (\eta_{3,0} + \eta_{1,2})^2 + (\eta_{0,3} + \eta_{2,1})^2 \\
\phi_5 =& (\eta_{3,0} - 3\eta_{1,2})(\eta_{3,0} + \eta_{1,2})((\eta_{3,0} + \eta_{1,2})^2 - 3(\eta_{2,1} + \eta_{0,3})^2) + \\
& (3\eta_{2,1} - \eta_{0,3})(\eta_{0,3} + \eta_{2,1})(3(\eta_{3,0} + \eta_{1,2})^2 - (\eta_{2,1} + \eta_{0,3})^2) \\
\phi_6 =& (\eta_{2,0} - \eta_{0,2})((\eta_{3,0} + \eta_{1,2})^2 - (\eta_{2,1} + \eta_{0,3})^2) + \\
& 4\eta_{1,1}(\eta_{3,0} + \eta_{1,2})(\eta_{2,1} + \eta_{0,3}) \\
\phi_7 =& (3\eta_{2,1} - \eta_{0,3})(\eta_{3,0} + \eta_{1,2})((\eta_{3,0} + \eta_{1,2})^2 - 3(\eta_{2,1} + \eta_{0,3})^2) + \\
& (3\eta_{1,2} - \eta_{3,0})(\eta_{0,3} + \eta_{2,1})(3(\eta_{3,0} + \eta_{1,2})^2 - (\eta_{2,1} + \eta_{0,3})^2)
\end{aligned} \tag{6.29}$$

Later it has been developed methods for generating moment invariants of arbitrary order [57, 24]. However using moments of very high order will represent very specific features in our data and with a huge variation between samples of the same class, we will focus on finding more general patterns. We use the inverse image, $1023 - f(x, y)$, so the outside mask will not be considered and we get a more direct measure of DNA distribution.

When investigating the general effect of the Hu moments we found that they all had values very close to zero. If the DNA were uniformly distributed in the cell and the shape was close to elliptic we would expect that ϕ_3 to ϕ_7 would be zero. We see that the effect of the DNA distribution is very small. From figure 6.19 we can se that ϕ_1 do indeed have an effect and can separate K4 cells to a certain degree, but this effect stems primarily from the morphological aspect, and is very related to *circularity*, *eccentricity* and *max diameter*. With a uniform DNA distribution and a shape close to elliptic, we would have $\phi_1 = \frac{1}{4\pi}(\frac{a}{b} + \frac{b}{a})$, where a and be are the major and minor axes, and this is indeed close to what we observed.

We tried to do a classification based on each of the Hu moments. From Table 6.5, we can see that all the features, except from ϕ_1, showed no effect.

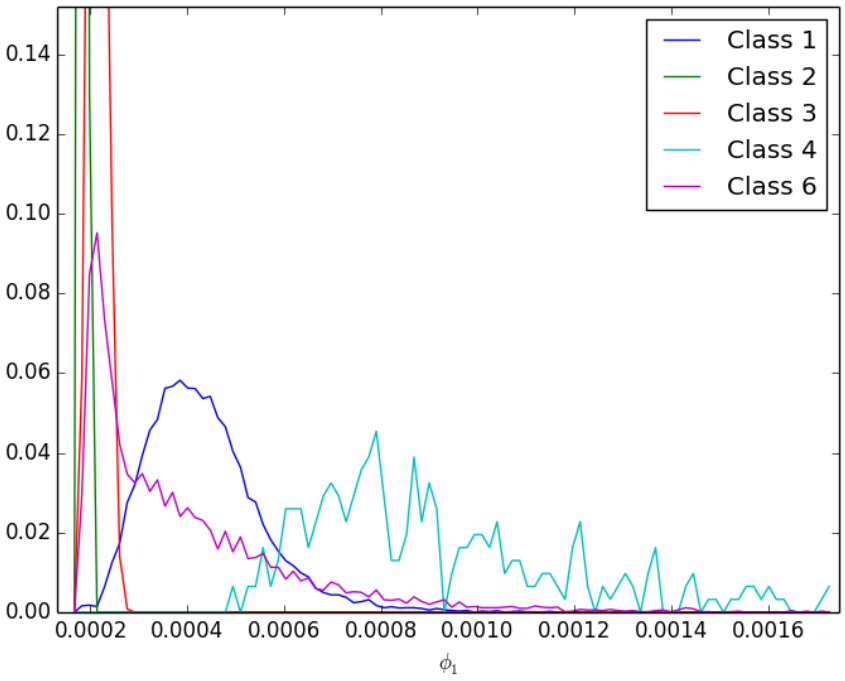


Figure 6.18

Figure 6.19: Histogram over the first Hu moment ϕ_1. The y-range of K2 and K3 are cut, as they are irrelevant in this setting.

We also tested two slightly more intuitive measures, but related to the moment based approach. We calculated the distance from the mass middle of the mask to the mass middle of the IOD distribution, and called this features *balance of IOD*. We also investigated the variance in distance from the mass middle of the IOD distribution and the IOD of each pixel. For each pixel we calculated the squared distance to the mass middle and calculated a mean, weighted by the IOD of that pixel. Both features gave quite similar results to the ones obtained by Hu moments. The balance of IOD seemed to give somewhat noisy results, but for K4 the probability that the IOD is centered far away from the cell center is greater in the elongated cells. Our variance measure obviously also gave a very high value for K4 cells, as they tend to have DNA distributed far away from the center. We kept the balance of IOD measure, because it seemed to be able to separate some K6 cells, over-segmented, overlapping or with foreign objects. We finally tested Zernike moments [79], where we projected our cell images onto a set of the 15 first Zernike basis functions. We used the center of mass of the cell mask as center and a radius equal to the the average of the major and minor axis of the cell. Classification of a part of the M51 dataset showed that each of the 15 features had between 25% and 55% CCR by them self, and combined either with gradient boosting or linear discriminant analysis

Table 6.5: Classification result for each separate Hu moment on the M51 dataset.

	CCR
ϕ_1	63.8
ϕ_2	25.0
ϕ_3	25.0
ϕ_4	25.0
ϕ_5	25.0
ϕ_6	25.0
ϕ_7	25.0

only gave a total CCR of about 58%.

From our measures it seems to be little information regarding the general distribution of DNA in the cell, that is strongly related to the class labels. The main reason for the lack of relevant information is most probably due to the way monolayers are prepared. As the images are projections of 3-D nuclei. If for example some cells tend to have most DNA far from the cell center, this DNA will still be projected at the center in our 2-D image. In this scenario we would still expect more DNA to be found along the edge of our projected cell, but the effect would be much smaller. Of course there could also be a tendency that DNA clustered in some part the cell, so the balance could be relevant, but even this effect could be weakened by the projection. We also have to note that all our methods have depended heavily on the cell shape, and therefore the information may be buried in noise from that shape. Perhaps we could still find relevant information either by using a peel-off-scanning, similar to the methods applied in [60], or by for example using a geometric transform to standardize the shape, and then for example use Zernike moments of the transformed data.

6.5 Granularity

Granulometry features can be used as a texture measures and detect grainy patters, they measuring how much is removed by doing morphological gray-level openings for different sizes of structuring elements [16, 17]. This attempt were not very successful and we found that a very simple feature gave the same separability, namely the sum of the Laplacian of the image. We only use the central 70% of the cell nuclei to calculate the features, so the cell boundary will not affect the result. The laplacian $\Delta^2 f$ is obtained by convolving the image with the 2D kernel:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{6.30}$$

Our feature *sum of Laplace* is then simply the sum of $\Delta^2 f$ over the region corresponding to the central 70% of the nuclei. The sum of the Laplacian could also be used as a focus measure.

6.6 Summary

In this chapter we evaluated the discriminatory effect of different features. We tried to encompass as many aspects of the cell images as possible, but still be restrictive in our selection of features. We found that morphological features provide quite good discrimination between the classes K1-K4. Many of these features is related to either cell area or the eccentricity of a cell. We did not include eccentricity, minor axis length or the skeleton length as features, as they proved more susceptible to noise, but still very related to many of the other included features.

The GLCM approaches seemed to capture more information than features only based on the gray-level histogram. The location or more broad distribution of DNA in a cell nuclei seemed to be more or less unrelated to the class labels. For the classification we chose to use: IOD, circularity, maximum diameter, eccentricity, area, perimeter, convex hull deficiency, concavity depth, bending energy, longest flat line, jaggedness, symmetry, adaptive GLCM features, GLCM contrast for $d = 12$, GLCM ASM for $d = 4$, mean gray-level, gray-level variance, gray-level skewness, gray-level entropy, Hu moment ϕ_1 and balance of IOD.

Many of these features are closely related, but as the classification algorithm employ a rough feature selection procedure, correlated features is less of a concern.

The computation of the features is primarily done i Python with the use of Numpy [83]. We also use OpenCV [45] for 2D filtering, contour detection, fitting ellipses and morphological operations. For visualizations we use MatplotLib [43] and for the 3D visualization of the GLCMs, we used Mayavi [72].

Chapter 7

Classification

In this chapter we will give a short discussion concerning our choice of classification model. We will also provide some details of our chosen methods of classification, and discuss some of the parameters we can tune in order to adapt the models to our problem and the effects of adjusting these parameters. Our interpretation is primarily based on the descriptions in [8, 29, 30, 66] and the chapters 9, 10, 15 and 16 in the book [39].

7.1 Choosing a Method of Classification

The aim of our classification method is to find a function $f(\boldsymbol{x}_i)$ that predicts a variable y_i from a vector $\boldsymbol{x}_i$. For our application this means that we want to find a function that estimates the class, based on a number of features, calculated from the cell images. We estimate f with the help of a set of data were we know both $\boldsymbol{x}_i$ and y_i, in other words *supervised learning*. The problem is then to choose a method of estimating f, so that it predicts y_i as precisely as possible, also for a new set of samples $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_N, y_N)\}$. The method that works best is obviously dependent upon the dataset, but it often comes down to how complex pattern a model should allow. Throughout the literature there have been suggested a wide range of methods, and we will discuss some of them, based on different aspects of or project.

7.1.1 Intuitive or Black Box

There are many examples of advanced classification models that create much more complex decision boundaries than humans could do. A Neural-Net (NN) or a Support Vector Machine (SVM) [85] can create arbitrary decision boundaries in a high-dimensional space. The problem is that it is also very difficult to understand for humans, how these models work, why misclassifications happen, and what could be changed in a model to improve the result. They are what we could call black box approaches, as we put data in, get a classification result out, but have little understanding about the decision process. In some applications human understanding is not of high priority. If there is little information about the underlying mechanisms, the relationship between the features is very complicated and we are not looking for a better understanding of the decision

mechanisms, a black box approach could be a good solution.

When we want to filter our data sets, we know that the training labels can be unreliable. With a black box approach it would be very hard to dispute the labels since we do not know how the decision was made. This could of course be improved by creating a very good training set, but that would be a very extensive procedure and not in our power to do, as we lack the expertise to create such a set. Another important issue that makes the use of a black box model problematic is the importance of the outliers. In our case we cannot simply remove the outliers as they may be important as prognostic information. With a black box approach it would be very hard to make sure that those cell images were kept. In our filtering problem we also have quite good information about the cells we want to remove, an important part of the problem is really where the thresholds should be set. We therefore decided to first attempt to use simple thresholding to filter out K6 cells. This means that the strictness and relative importance of different features can easily be chosen by experts, depending on different situations. We also attempted to make a very specific set of features directed towards some of the categories of cells contained in the K6 class, described in section 2.4.6, as this makes the procedure as intuitive as possible. An important aspect of this is obviously also that it can ultimately result in a solution that can outperform the manual filtering. This approach is described in detail in chapter 5.

Using a simple classification procedure will most likely result in a lower estimated accuracy, but the value of an easily interpretable model may outweigh the value of lower accuracy for the filtration task.

The classification of the cell types is a quite different situation. The labeling for this problem is more reliable and at least without education in pathology or molecular biology we can probably not make much sense of the underlying mechanisms. If we find that a certain class tends to have a certain size, shape or texture, we do not have the expertise to use or dispute that information. We could for example not say whether a feature really is representable for a class or if this is just a coincidence, based on such information. When we additionally seem to need a quite complex classifier to find an appropriate decision boundary, this makes a better case for a more black box approach.

7.1.2 Scaling

In classification methods like Maximum Likelihood, Nearest Neighbor and Support Vector Machine [14, 73, 78], the effect of the relative scaling between features is potentially huge. Classification methods that use Euclidean distance to find the decision boundary will all be affected by the relative scale of features. The features with a large scale would give a much higher contribution to the distance, which gives them more influence on the finial result. The problem is that the initial scaling of the features is quite arbitrary in our case.

An easy way tackle the scaling problem could be to scale all the features to a given range or to scale it to a given standard deviation. The problem with these approaches is the outliers. Features where outliers have extreme values will get very compressed and the distance between the inliers on this dimension could end up very small. Then we are essentially left with features that only contribute to separating out the outliers, but have little effect on the inliers. This problem is further complicated when we use a feature to help separating

multiple classes. For example area is very helpful in separating K1, K3 and K3. K2 and K3 have quite small average difference in area between them, but as their variation in area is also very small, they can still be quite well separated based on this features. K1 cells on the other hand are averagely much larger than K2 and K3, but the variation in cell size is huge. If we scale this feature based on the combined standard deviation, the difference between K2 and K3 will be very small in Euclidean distance as the total standard deviation is huge. Therefore area may count very little compared to other features, even though it may be the feature with best separation.

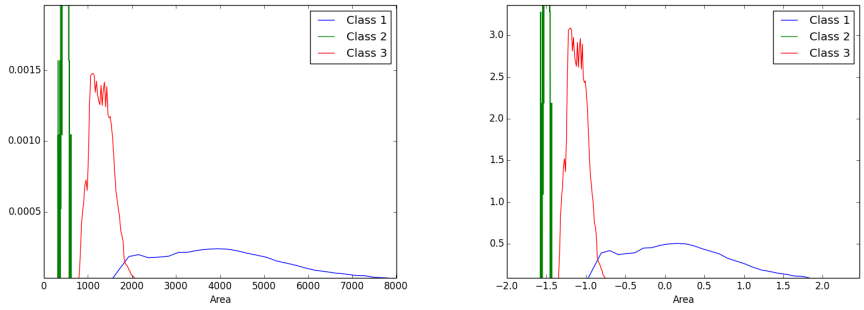


Figure 7.1: A histogram over the area *for K1, K2 and K3. All the samples are taken from the M51 data-set. To the left we can see the unscaled distribution, while on the right is normalized to have mean of 0 and standard deviation of 1. The two distributions obviously looks similar, but we can se that the difference between K2 and K3 is only about 0.5 standard deviations on average. This means that this difference may have very little influence on for example an optimal margin classifier such as SVM or a KNN classifier. Still we can see that the two classes are in fact perfectly separated on this feature.*

We could instead utilize the relative scaling to intentionally influence the importance of different features. We actually tested this with an evolutionary approach; a search for the optimal scaling of the features. Unfortunately we found that this easily led to an overfitting to the different test set. Each training set gave very different scaling and as this process was immensely time consuming we discarded this approach. Instead we focused our attention on classification methods that worked independent of the relative scaling of the features and only focus on the separability.

7.1.3 Features

Features are the basis for the classification and if two samples of different class have exactly the same feature values, there is no way for the classification model of separating the two. In other words, we need to use enough features to capture the differences between the classes. The ability to separate classes increase with the number of features, but for each feature added we will also add noise, as we have no features that measures differences between classes directly. A feature may contribute little or no information to a model, either because that information is already accounted for by other features or the feature is irrelevant

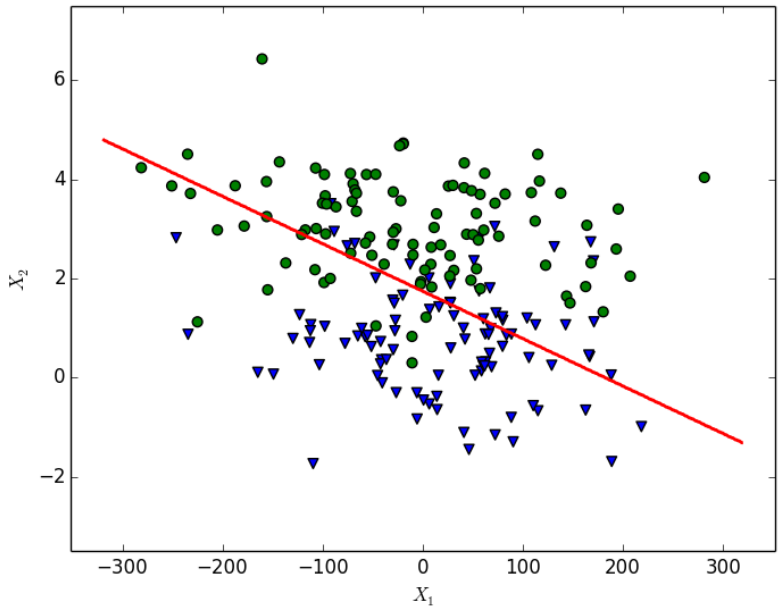


Figure 7.2: An artificial dataset with two groups, randomly generated from two gaussian distributions. Both groups are generated from distributions with a mean of 1 and a standard deviation of 100 on the x-axis. On the y-axis both distributions have a standard deviation of 1, but the distribution that the green circles are generated from have a mean of 3, while the blue triangles are generated from a distribution with a mean of 1. The red line is the decision boundary found by a SVM with a linear kernel and $C = 1$.

to the class label. If we add irrelevant features, we add noise but no information. With increased noise our model can easily fit that noise, an loose generalizability. Such a scenario, where the model fit to the noise is called *overfitting*.

With higher dimensional feature-space, on can end up in a scenario where very few samples decide the decision boundary in a region, as the size of the features-space increase dramatically for each added feature. This effect is usually referred to as *the curse of dimensionality*. In figure 7.2 we illustrate how a few samples can affect the decision boundary, when an irrelevant feature is added to the model. From this figure we can also see how the irrelevant feature is used even though the true distribution for that feature only contain noise.

Bias Versus Variance

The balance between the getting a good discrimination and the ability to generalize from a restricted training-set, can be viewed in terms of *bias* and *variance*. For a very restricted model, with small degree of freedom, it may be parts of the data that is inseparable as we lack some information for describing the class differences. This remaining part of errors that we cannot improve regardless of how perfect a training-set we have, can be called *bias*. A very complex model,

with high degree of freedom, may on the other hand include more information, and therefore have the opportunity to come much closer to the perfect result. It will also be able to fit the individual differences in the training-set better and therefore the result will vary much more depending on the training-set. The *variance* of a model is then how much the results vary, depending on a given training-set. The total error of a model will then be a sum of the bias and the variance.

Feature Selection

To keep a model from falling under the cures of dimensionality, it is often applied some sort of feature selection procedure. These procedures can be view as a sort of simplified classifiers. If the classifiers are restricted in some way, they will only have opportunity to fit to the most discriminatory effects or features, and these strongest effects are less likely to be caused by a coincidence.

A simple way of feature selection, is to simply evaluate the discriminatory effect of each feature individually, judged on some criterion, and then choose the best scoring features. This strategy is a great in order to avoid overfitting, but it completely ignore *interaction effects* between features. A feature may have little or no discriminatory effect by itself, but have a strong effect in combination with other features. In figure 7.3 we illustrate how two features can have no real discriminatory effect by themselves, but still have a great effect when combined. When only considering features individually one will also face a problem that often prove even worse, namely that many features of the features may correlate and therefore provide little additional information.

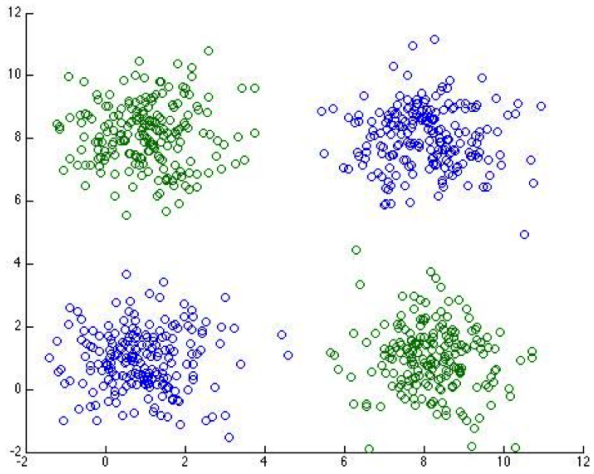


Figure 7.3: An extreme example of interaction effect, where the features have as good as no discriminatory effect by themselves, but together can provide perfect discrimination.

Another approach is to consider the discriminatory power from subsets of features, where the extreme approach is to use the *brute force* solution and evaluate the discriminatory power of all possible subsets of features. To best separate the training-set, the best subset would of course be to use all features.

Therefore it is common to decide the decision boundary using a *training-set*, and evaluate the sets of features using a *validation-set*. The brute force solution will find the optimal set of features for a validation-set for a given a training-set, but is very computationally intensive. More common procedures is therefore to use iterative *forward* or *backward selection*, where the feature that provide the best additional separation is added or the feature that provide the least separation is removed. This is done iteratively until either a certain number of features is added or removed, or separability of the validation set is no longer improving. By using one of these iterative techniques, the problem of picking only highly correlated features may be avoided. As only one feature will be removed or added at one time, interaction effects like the one illustrated in 7.3, may still easily be missed. The situation where the scope of a greedy approach is to small to capture an effect, and therefore choose a suboptimal solution, can be referred to as a *nesting problem*. In the literature there have been suggested wide variety of solutions to this problem, for example by *stepwise forward-backward selection*, where it is possible to both add and remove features for each iteration, [35]. How nested features may affect gradient boosting and random forests will be discussed in section 7.5.

Regardless of features selection procedure, using many features will still increase the probability that irrelevant features are chosen by coincidence. With a training- and validation-set it means that the feature would have to have similar effect on both sets by a coincidence. Schulerud and Albregtsen demonstrated in a simulation study [76] how the probability for selection the relevant features decreased with the number of added feature candidates, and how the added features demanded more training-samples in order to average out the effect of the added noise.

All this suggest that we have to limit the set we include in our model at the very beginning, but if we are excluding features by merely looking at the training-data we are not necessarily doing any better than a feature selection algorithm and are exposed to the same effects. What we can do is to avoid including features that are obviously correlated as this would make us exposed to all the dangers of added features, but probably provide little additional information.

7.2 Classifictation and Regression Tree

We decided to base our classification on a *decision tree*. For our purpose this gives us several advantages. The whole scaling process becomes irrelevant, as each step in the training procedure sets a threshold that does not depend on distances in features space. Feature selection also becomes less essential with this tree based approach, since a greedy selection procedure is already built into the algorithm. For illustration purposes we trained a simple classification tree in figure 7.4.

With a decision tree we group samples together into regions, by their values in the vector $\boldsymbol{x}$. All samples in one such region will get the same value for the estimation of y. So for a given tree we have a number of regions $R_1, R_2, ..., R_J$, and we have a estimated value for each region $\gamma_1, \gamma_2, ..., \gamma_J$. The tree is then

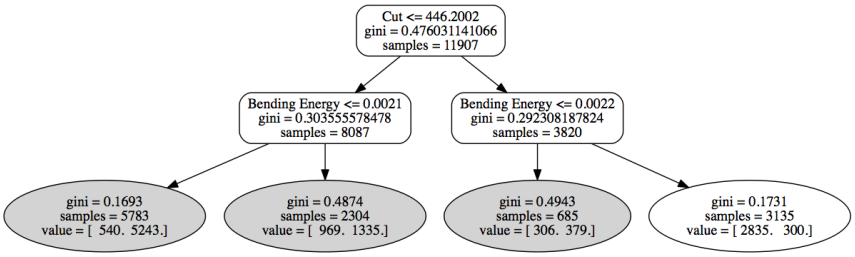


Figure 7.4: An example of a classification tree for filtering out K6 cells. The depth of the tree is restricted to two, just to limit the size of the figure. The elliptic node are the leaf nodes and determine the finial class of a sample. A sample ending up in a node with a gray background will be kept, while a sample ending up in a white node will be classified as a K6 cell and therefore removed.

defined as a function of $\boldsymbol{x}_i$

$$T(\boldsymbol{x}_i;\theta) = \sum_{m=1}^{J} \gamma_m I(\boldsymbol{x}_i \in R_j, \tag{7.1}$$

where θ is the set of p all the regions and values $\theta = R_j, \gamma_j{}_1^J$. For regression the estimate is just the value of $T(\boldsymbol{x}_i;\theta)$ directly. For binomial classification it is common to use $f(\boldsymbol{x}_i) = sign(T(\boldsymbol{x}_i;\theta))$, for multinomial classification we create one tree for each class $T_1, T_2, ..., T_K$ and our estimate is then

$$f(\boldsymbol{x}_i) = \arg\max_k T_k(\boldsymbol{x}_i;\theta). \tag{7.2}$$

7.2.1 Splitting the Population

A decision tree is created by iteratively splitting a training set into subsets, until each subset contains only one class or the subset size is appropriately small. Each leaf node will then represent one class, given by a voting strategy. The challenge, when training a decision tree, is therefore to find the best thresholds for splitting the population. There are several different variants of decision trees [71], but we focused "Classification and Regression Trees" suggested by Breiman et al. [66].

To find optimal thresholds can obviously be time consuming, so the search is simplified by using a greedy strategy. For each step we can simply search through all variables and possible splits and decide on the best one. To decide the best split we need some way of evaluating the quality of the split quality. For both classification and regression purposes there exists a variety of such measures; termed impurity measures for classification and criterion of minimization for regression problems.

Classification

Perhaps the simplest measure for classification is the misclassification rate. If $p_{m,k}$ is the proportion of samples of class k in a subpopulation m, then the

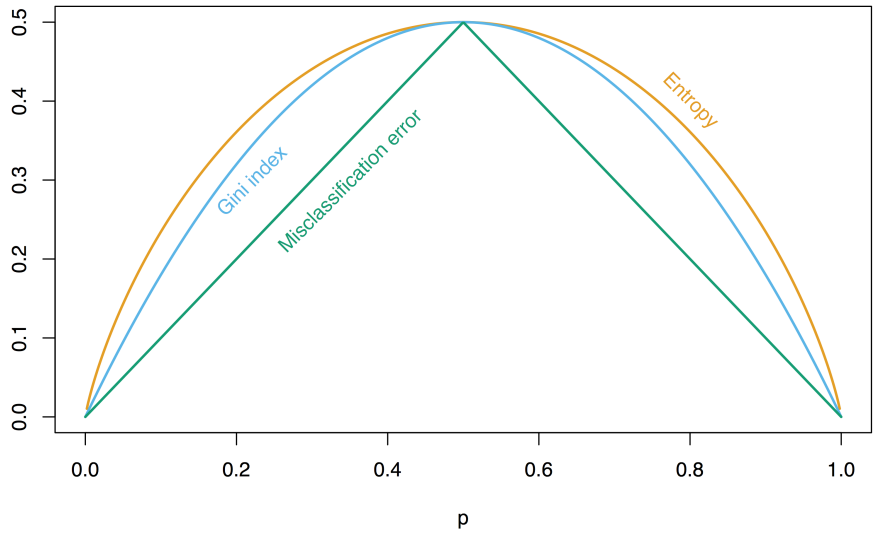


Figure 7.5: Displaying how different values of p is valued with different impurity functions. Copied from the book [39, pp.309].

misclassification rate can be calculated as the proportion of samples in the population, that is not in the most common class. The misclassification rate can then simply be expressed as,

$$Q_m = 1 - \max_k p_{m,k}. \tag{7.3}$$

The split with the highest quality will be the one with the lowest sum of (7.3) for each of the subpopulations, lets call them L and R,

$$\varphi = Q_L + Q_R. \tag{7.4}$$

The Gini Index (7.5) and Entropy (7.6) are other impurity measures.

$$Q_m = \sum_k p_{m,k}(1 - p_{m,k}) \tag{7.5}$$

$$Q_m = \sum_k p_{mk} \log p_{mk}. \tag{7.6}$$

The main difference from the misclassification rate (7.3) is that entropy and the Gini index will value cleaner splits relatively higher. By cleaner splits we mean splits where one of the subpopulations have a very low misclassification rate.

Minimizing the entropy can also be interpreted as choosing the variable and threshold that contains the most information. If the class labels were not assigned by majority vote, but instead were assigned with probability $p_{m,k}$ the Gini index would be the estimated misclassification rate. If you are at a non-leaf node, the probability of being classified correctly is indeed related to the class distributions, $p_{m,k}$. The Gini index is sensible as the impurity measures are primarily relevant in non-leaf nodes. In [91], Zambon et al. compared the

effects of different impurity measures. For their data, the Gini index gave the best result, but they also emphasize that the effects were small.

Regression

For regression problems we do not have the same concerns about the quality measure of a split and we can simply use the mean squared error. If R_m is a region from a subpopulation in a split, then we have the quality measure of a the region

$$Q_m = \sum_{X_i \in R_m} (y_i - f(x_i))^2. \tag{7.7}$$

The quality measure of the split is then calculated the same way as for a classification problem with equation (7.4), as the sum of the quality measures for the two regions.

7.2.2 Pruning the Tree

The splitting process could continue until a node only contained one sample. This would make the model fit the training set perfectly, with no classification error. In most circumstances such a perfect fit of the training set is not wanted, as it would almost certainly lead to overfitting and therefore poor generalization. We would like to reduce the complexity to avoid overfitting, but not still keep the most relevant information. We could choose to stop the splitting if the gain in accuracy is not high enough. The problem with this approach is that one bad split, may give great splits further down the tree. In other words, the problem of balancing cannot be solved by a stopping rule alone. A better alternative is to always split to a given subset size and then prune the tree afterwards.

For pruning procedure we use what is termed *minimal cost-complexity pruning*, presented in [66]. The idea is to first grow a large tree, which we call T_{max}, and then find the best tree $T \subset T_{max}$, where T can be any subtree of T_{max}. N_m is the number of training samples in the subtree T_m and $|T|$ is the number of leaf nodes in T. A cost complexity criterion is then defined as

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T|. \tag{7.8}$$

We then want to find a subtree $T \subseteq T_{max}$, for a given α that minimize the cost function $C_\alpha(T)$. The alpha becomes parameter for adjusting the balance of a complex tree with high bias or a small tree with high variance. α decides to what extent you want to take the tree size into account. $\alpha = 0$ will result in T_{max} as the impurity measure will get better or equal for each split. To find the best possible value of α we can use a grid-search in combination with cross-validation, as with a separate validation we avoid the problem that a bigger tree is always better [39].

7.3 Boosting

Boosting is a very popular machine learning technique and is used in many fields of research [15, 18]. A classification algorithm based on the boosting principle

often have strong predictive power and is relatively easy to use as an out-of-the-box classifier. It is a way of combining multiple "weaker" classifiers into a more powerful model. There are other methods for combining multiple classifiers into a stronger learner, but boosting stands out in the way in which it changes the distribution in the training set before training a new classifier. The distribution change is created by weighting the wrongly classified training samples more than the correct classified ones.

For our project boosting proves a convenient way of increasing the predictive power of decision trees, while still keeping the main advantages in that we do not need any scaling and that we have a built-in method of feature selection.

As for other classification methods we have a set of training samples $(\boldsymbol{x}_i, y_i)$ for $i = 1, 2, ..., N$ and we want to find a function f, that maps $\boldsymbol{x}_i$ to y_i,

$$f(\boldsymbol{x}) = y.$$

For a two-class classification problem, all samples with a negative value of f is classified to one class, and those with a positive value is classified to the other. For a multinomial classification problem with K number of classes on the other hand, we fit K functions f_k, on for each class. We find the label a sample to the class with the highest function value,

$$G(\boldsymbol{x}) = \arg\max_k f_k(\boldsymbol{x}).$$

We describe f as sum of basis functions

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m), \tag{7.9}$$

and boosting is a way of fitting the basis functions. M is the number of basis functions, so $m = 1, 2, ..., M$ and β_m and γ_m are the coefficients to be fitted. They are fitted with the help of a loss function L,

$$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)). \tag{7.10}$$

This is a sum of the loss function over all the training samples, and N is the number of samples in the set.

7.3.1 Loss Functions

For a boosting algorithm to work, it is essential to value misclassified samples more than correctly classified samples. The loss function determine how those samples are weighted as a function of the *margin*. By *margin* we mean the how "far" the sample where from being labelled to another class, if the sample were misclassified the margin will be negative. With a two class classification problem we calculate the margin simply as $y \cdot f(\boldsymbol{x})$, where y is either 1 or -1. For multinomial classification we still want to get a measure of how wrong or how correct a sample were classified. In this situation we calculate the margin in the exact same way, but we only use the the function f, corresponding to the correct class y of the sample.

Two of the most common loss functions are *exponential* and *binomial deviance*. The exponential loss function is a built-in part of the widely used AdaBoost algorithm, developed by Freud and Schapire [26]. The details concerning how the loss function works were discovered later, with a slightly alternate interpretation of boosting. Friedman et al. [29] showed that the AdaBoost algorithm, can be interpreted as a additive model using the following loss function

$$L(y, f(\boldsymbol{x}) = exp(-yf(\boldsymbol{x})), \tag{7.11}$$

and that the expansion produced by AdaBoost is estimating half the log-odds of

$$P(Y = 1 \mid \boldsymbol{x}),$$

by applying the algorithm to the population joint distribution. The *binomial deviance* loss function gives the exact same result for the population joint distribution, but they differ when applied to a limited training set. The binomial deviance function

$$L(y, f(\boldsymbol{x}) = log(1 + e^{-2yf(\boldsymbol{x})}), \tag{7.12}$$

can also be adapted to a multinomial problem. We first map f to a probability function

$$p_k(\boldsymbol{x}) = \frac{e^{f_k(\boldsymbol{x})}}{\sum_{i=1}^{K} e^{f_i(\boldsymbol{x})}} \tag{7.13}$$

and get the *multinomial deviance* loss function,

$$L(y, f_k(\boldsymbol{x})) = -\log p_y(\boldsymbol{x}), \tag{7.14}$$

if y corresponds to k; otherwise we need a mapping function.

The main effect of these loss functions is that exponential loss, weights the misclassified samples exponentially as a function of the margin. This means that samples that are classified very wrong get a very high priority in the next iteration of the algorithm. Deviance loss on the other hand weight the misclassified samples linearly as a function of the margin. This means that extreme mistakes are weighted relatively less. Both functions are quite similar in how the weight the correctly classified samples for the next iteration. The different weighing for these and other functions are illustrated in figure 7.6.

A problem with the the exponential loss is that it can give a somewhat unstable result. Its high weighting on large negative margin will make the algorithm vulnerable to outliers and noise in the data. In fact an empirical study published by R. Maclin et al. [53] confirms that AdaBoost can yield unstable result for noisy data. Sano et al. demonstrate how AdaBoost is especially vulnerable to miss-specification of labels and that only a 2% miss-specification rate can give a drastic decrease in performance [74] (see figure 7.7).

7.3.2 Gradient Boosted Trees

One of the main reasons for choosing an exponential loss function is convenience, but for the current study it might be beneficial to use the slightly more robust deviance loss. To implement that loss function we need a slimly more

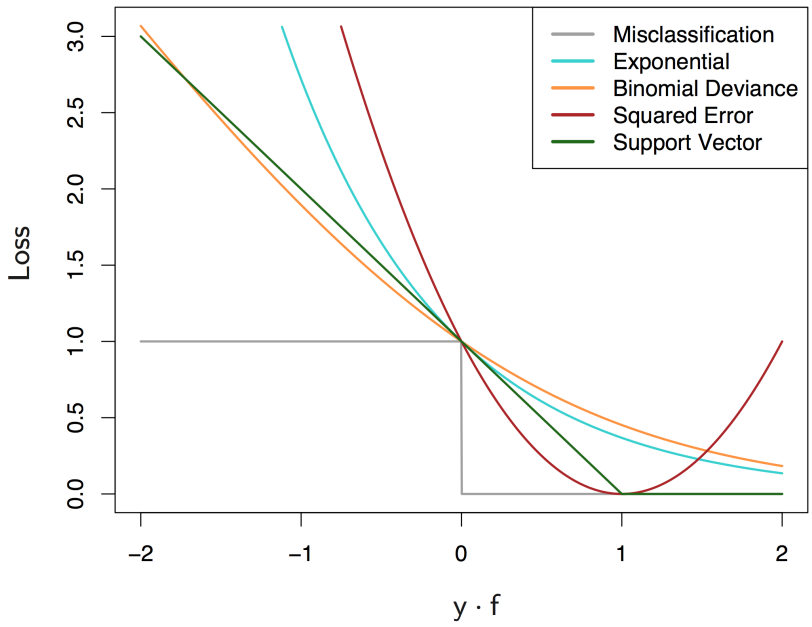
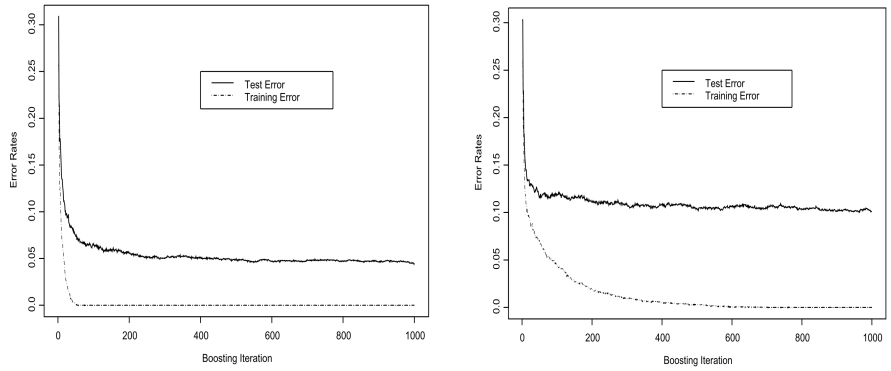


Figure 7.6: An illustration of how the negative margin is weighted much higher with exponential loss. Source [39].



(a) *(b)*

Figure 7.7: Plot (a) show the training and test error of standard AdaBoost algorithm without mislabeled data. Plot (b) show training and test error for the same algorithm, but with 2% mislabeled data. Sano et al. demonstrates with this figure the drastic decrease in performance of AdaBoost with a small percentage of mislabeling. Copied from [74].

cumbersome, but more general approach. A general boosting tree model can be described as the sum of the trees,

$$f_M(x) = \sum_{m=1}^{M} T(\boldsymbol{x}; \Theta_m). \tag{7.15}$$

When training the model a tree is fitted, for each iteration, by estimating its parameters $\hat{\Theta}_m$. $\hat{\Theta}_m$ is found by

$$\hat{\Theta}_m = \arg\min_{\Theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(\boldsymbol{x}_i) + T(\boldsymbol{x}_i; \Theta_m)) \tag{7.16}$$

With an exponential loss function this minimization problem is easily solved, without the need for the Gradient Boosting technique and steepest decent. The main advantage of the Gradient Boosting technique is the ability to adapt the algorithms to any differentiable loss function quite effortlessly [30]. Indeed, for the deviance loss function, the minimization problem (7.16) cannot be solved readily, and gradient boosting proves useful.

The idea of Freidman's approach [30] is to find the gradient of the loss function $\boldsymbol{g}_m$, with respect to $\boldsymbol{f}$ at the points of the samples,

$$\begin{aligned}
\boldsymbol{f}_m &= [f_m(x_1), f_m(x_2), .., f_m(x_N)], \\
\boldsymbol{g}_m &= \left[\frac{\partial L(y_1, f_m(x_1))}{\partial f_m(x_1)}, \frac{\partial L(y_2, f_m(x_2))}{\partial f(x_2)}, ..., \frac{\partial L(y_N, f_m(x_N))}{\partial f_m(x_N)} \right].
\end{aligned}$$

Then we could "move along" the negative gradient, as we know that this is the shortest way to reduce our loss function. In other words, a way to decrease L, with the least change in $\boldsymbol{f}$. This means that we can update $\boldsymbol{f}$ iteratively, by adding the scaled negative gradient,

$$\boldsymbol{f}_m = \boldsymbol{f}_{m-1} - p_m \boldsymbol{g}_{m-1},$$

where p_m is the scaling that determine how far we move along the gradient. The problem now, is that the vector $\boldsymbol{f}$ and the gradient $\boldsymbol{g}$ are only defined at the sample points, $\boldsymbol{x}_i$, and we obviously want to build a classifier that can label new samples. An important idea proposed in [30], is that we can fit a regression tree to the gradient, the same way that we would normally fit f to the sample points. Then we can easily fit the tree by least squares

$$\hat{\Theta}_m = \arg\min_{\Theta} \sum_{i=1}^{N} (-g_{i,m} - T(x_i; \Theta))^2. \tag{7.17}$$

In the proposed algorithm they do not actually move exactly along the gradient. The gradient is only used to find the regions $R_{j,m}$, in other words what part of our function, f, that move in the same direction. A further adjustment to is made to the direction, when the the value of each region is set, by minimizing a loss function for each region in the regression tree,

$$\gamma_{j,m} = \arg\min_{\gamma} \sum_{x_i \in R_{j,m}} L(y_i, f_{m-1}(x_i) + \gamma),$$

where $\gamma_{j,m}$ is the estimated value of f_m in region $R_{j,m}$. The estimated function at iteration m is then

$$f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{j,m} I(x \in R_{j,m}),$$

where J_m is the number of regions for this iteration. We now have the complete approach:

1. We start of with and initial guess $\boldsymbol{f}_0$
2. fit a tree to the negative gradient with (7.17)
3. do a line search for values of γ for each region, to minimize the loss function
4. update our current function f, with 7.3.2

Step 2-4 is repeated for a given number of iterations M, and our final model is f_m. This approach can now be used for a number of loss functions, as long as they are differentiable. The gradient for the multinomial deviance loss (7.14)

$$\frac{\partial L(y_i, f_k(x_i))}{\partial f_k(x_i)} = \begin{cases} 1 - p_k(x_i) & y_i = k \\ -p_k(x_i) & y_i \neq k \end{cases} \tag{7.18}$$

To get the best possible result, and adapt this method to our purpose we have to decide a number of meta parameters.

7.3.3 Important Parameters

There are several ways of tuning the algorithm to find the right balance between bias and variance for a given application. The two most obvious is the number of trees created and the depth of trees. As each iteration is weighting the errors the most, the classification should adapt more to the training set for each tree added. By restricting the number of iterations we can keep the model from overfitting to our data. In the same way, we can restrict the depth of the tree, and prevent it from growing too complex. A complex tree can adapt to noise in the data and therefore get overfit. In other words, they are both parameters for regulating the bias versus variance trade-off.

In [39], they point out that the tree-depth are related to interaction effects in the data. Each tree can tap into interaction effects with degree one lower than the tree-depth. It is therefore suggested that the tree depth should be related to the expected interactions in the data. Hastie et al. [39, pp.361–363] and Friedman [30] found that a tree-depth of 2 (only one split and two terminal nodes), work best in some situations, and that a tree depth of 4-8, works well in most situations. With a depth of 2, a tree can only account for main effects of the features, so the combined model cannot solve XOR-type problems. So for the appropriate data this could be a good way of preventing overfitting. Smaller trees will also give more of the power to the gradient of the loss function. In the extreme case were each training sample have its own terminal node, the gradient would have no effect, as its only effect is in how to group samples together. Friedman is there for suggesting to keep the trees small and regulate overfitting by adjusting the number of iterations [30]. This can be done by

applying a validation set under the training procedure, and stop iterating when the validation error flattens out or increase. Another parameter for regulating the complexity of the fit, is the shrinkage parameter v. We can control the rate of learning by scaling the contribution of each tree by with a value $0 < v < 1$. The model can benefit from the smoother decision boundary created by many iterations, but with a downscaled contribution of each tree, the model can find a local optimum for the decision boundary very fast, and more iterations only contribute to overfitting. A lower v, can allow more iterations without overfitting the model, but this will of course increase the need for computational power. Friedman found with empirical experiments, that a low value of v and a high number of iterations can indeed be very beneficial in terms of testing error [30].

7.4 Random Forests

Random forests [8] is a very popular method for classification and regression. They can accomplish similar result as with boosting algorithms [39], and sometimes even outperform them [55, 88]. A fact adding to their popularity is that they are easy to understand, tune and parallelize. They are additionally much faster than boosting. In other words fast, easy and efficient.

The main concept of random forests relies on *bagging*. Bagging in terms of machine learning, means that you train multiple classifiers on special subsets of the training samples. The subsets are created by drawing N samples with replacement from the whole population of samples. The final classification result is then obtained by a majority vote among all the classifiers. Such a procedure helps to average out high variance of classifiers, to create a more robust classification model. This is an excellent strategy for high variance classifiers like classification tress.

A problem when using "bagged" classification trees is that the trees have a tendency to correlate. The same features are strongest across many different subset, and are therefore selected first. With high correlation among the trees, the high variance outcome persist. Breinman suggested to alleviate this problem with the introduction of random forests. The idea is then when the decision trees are trained, a subset of features is randomly selected and the node can only split based on that subset. In that way the trees are forced to choose different features, and in that way they are de-correlated. The correlation between the trees can be controlled by adjusting the size of the subset of features available for each node. The de-correlating effect is obviously zero when the subset is equal to the set of features. By reducing the number of features available at a node, the correlation can be decreased, but each tree also loose predictive power. It is therefore important to find the right balance between predictive power and correlation. Test indicates that smaller data set benefit from having very small subsets (1-4 features), while larger data had better effect of more features [8].

7.4.1 Important Parameters

The number variables considered at each tree node is one parameter we can adjust in order to adapt the algorithm to suit our needs. As with gradient boosted trees, we can restrict the number of trees in our model, but this has a slightly different effect for random forests. As the number of trees grows,

the model does not continue to get more overfit. As a result of this, adding more than a certain number of trees will have very little effect. In two examples provided by Hastie et al. [39, pp.589–591], they show that the test-error stopped improving after about 200 trees were added. To adjust the complexity of the decision boundary we therefore mainly have to adjust features considered at each node and the maximum depth of a tree. Breiman suggests to keep the tree complex with little or no restriction in terms of tree depth, reduce the number of features to a very small set and to use a huge set of trees [8].

7.5 A Nesting Problem

Friedman et al. describes boosting procedure as greedy *forward stagewise* fitting [29]. Only from this short description one can sense that the algorithm have a nesting problem. The selection of the first feature will influence the weighting of the samples and can therefore affect the selection of all other features. As the features are chosen greedily, it is easy to imagine how nested features like the ones presented in figure 7.3, will never be chosen, despite their combined discriminatory effect. Buja argue as a comment in [29], that the suboptimal feature selection is indeed why boosting is very robust against overfitting. Investigating the interaction between two features is in many ways similar to investigate a new feature. With the power of combinatorics we can easily see that there are enormous amounts of possible interactions even for quite small number of features. Since Holte illustrated that for most common datasets, there are no complex interactions [41], it may be that using elaborate feature selection schemes can do more harm than good, for many applications. The importance of feature interactions do of course depend on the feature generation process. If we use specific high-level features it is unlikely that features that have very little discriminatory effect individually, will have have a large effect in combination with other features.

For the Random Forest algorithm, the selection of the first features will not influence he selection of features in other trees. With semi-random feature selection it can pick features that appear bad when only the impurity measure is considered. As the discriminatory power of an individual feature still influence the selection, the effect of the randomness will be average out, an the algorithm is still left with a nesting problem. Using deep and pruned decision trees for the Random Forest might remedy the nesting problem slightly. With deep trees and small subsets of features to choose from, the probability of choosing a feature with little individual discriminatory power is relative large. When the tree is pruned, bottom up, the interacting features are likely to be kept.

7.6 Partial Dependence Plots

With complex classifiers as gradient boosting or random forests it could be difficult to interpret the model and get an impression of the decision boundary, when we use more then 3 features. For intuition we are limited to 3 dimensions, so we need some way of viewing the decision boundary for only a subset of the features at the time. So we want to investigate a subvector $\boldsymbol{x}_S$ of the input variable $\boldsymbol{x}$. We could of course hold all the values in the complement vector $\boldsymbol{x}_C$

constant. Then we find a cross section of the decision boundary $f(\boldsymbol{x})$, but the cross section does not necessarily provide any relevant information. However in the case where the effect of $\boldsymbol{x}_S$ are purely additive, the cross section would be accurate. So if we use a tree with maximum depth of 2, we could actually just do a cross section. To investigate more complex decision boundaries, with interaction effects, we could use the *partial dependence*, of $f(\boldsymbol{x})$ on X_S. This means that we average over the complement $\boldsymbol{x}_C$,

$$f_S(\boldsymbol{x}_S) = E_{\boldsymbol{x}_S} f(\boldsymbol{x}). \tag{7.19}$$

This gives us a way to find the average effect of a set of features. High partial dependence for some values, means that samples with those values are more likely to be classified to the given class, by the given classification model. In other words, samples in figure 8.1d, with a low circularity value are more likely to be classified to the K4 class.

7.7 Decison on Classification Model

We choose to use ensembles of decision trees, as ensembles like boosting or random forest, can give a better bias versus variance tradeoff than decision trees alone. Decision trees are suitable as *base learners* as we avoid the problem of scaling features and get a built-in feature selection. By restriction of the tree depth we can avoid problems with the curse of dimensionality, although irrelevant features will still reduce the probability of choosing the best subset of features.

From chapter 10 and 15 in [39], Hastie et al. found that gradient boosting generally outperformed random forests and other classification models. Caruana et al. [11] also found that boosted trees out performed random forests, but only when properly tuned. A comparison of [18] on the other hand, may indicate that bagging techniques such as random forests can outperform boosting algorithms under noisy conditions. As we could not find a definite choice for the best classification algorithm we will evaluate the performance of both a gradient boosting approach and a random forest. For the Classification Trees in the Random Forest we will use the Gini index, from equation (7.5), as the Q_m measure. The Gradient Bossted regression trees, will on the other hand use mean squared error, from equation (7.7), as the Q_m measure. The trees will not be pruned as this is not recommended for random forests [8] or gradient boosting [30].

For both random forest and gradient boosting we use the implementations provided i the open source project Scikit-learn [68].

Chapter 8

Results and Discussion

In principal result and discussion should be kept separate, so one can easily distinguish between facts and opinions. As our results are so extensive, we find it useful to interleave the two, to improve readability. The section concerning the feature value thresholding is inherently difficult to separate, as we do not use an objective thresholding technique, but visual inspection. We have stated our results clearly in tables, as to leave no doubt concerning the facts.

Our investigation on inter-observer reliability indicated that the K6 labels may be too unreliable to get a good result from a supervised learning approach. We therefore first present the results for only the cell-type classification between the classes K1-K4. Then we test out thresholds for detection K6 cells, and finally we test a supervised classification model on all classes.

As we indicated in Section 7.4, there are some studies that suggesting that *random forests* may be more robust to mislabeling, than boosting techniques, while other studies have found that gradient boosting are generally superior to random forests. We will therefore attempt a comparison between a gradient boosting and a random forest algorithm, as are the two machine learning technique that we find most suitable for this problem. For both algorithms we use the features presented in chapter 5 and chapter 6.

8.1 Cell-Type Classification

We first present the parameters that we found to be best for each of the classifiers. These are the parameters that we use throughout this study. We then move on to an investigation into the different features and their effects.

Finally we present the classification results. First we present results from a cross-validation on the training data, as it can be relevant to see the final results from the data sets that we have manually investigated. Next we present the classification results for the independent test-sets. And at the end of this section we discuss how these results can be interpreted.

Gradient Boosted Tree

For *gradient boosting*, the main parameters are *learning rate*, *subsampling*, the *number of estimators* and the restriction of the *max depth* of the regression trees. To find a good combination of the parameters we used a coarse grid

search. We used the same parameters for all the datasets as we do not consider the difference between them large enough to justify different parameters, instead we expect that it would only lead to an overfitted model. In table 8.1 we present the parameters found by the grid-search.

Table 8.1: Best parameters for the gradient boosting

Parameter name	value
Learning rate	0.02
Subsampling	1.0
Estimators	400
Max depth	3

Random Forest

For the random forest algorithm we tried to adjust the *number of estimators*, *max tree depth* and the number of *features considered* at each node. We found that quite restricted trees worked best, contrary to the suggestion of Breiman [8] suggestion of keeping the tree quite complex. The number of threes are still high, but we there was almost no improvement after 100 added trees, as show in table 8.2.

Table 8.2: Best parameters for the random forest

Parameter name	value
Features considered	3
Estimators	400
Max depth	6

8.1.1 Feature Importance

Even though our classification models are robust against correlated features, especially gradient boost, we did remove highly correlated features. One reason is obviously computational complexity, but the random forest algorithm can be affected as many similar features will increase the probability that one of those features are chosen, and in that way bias the selection process. A solution to this problem could obviously be to perform a method of feature selection. We could for example perform a brute-force solution or a floating search. Such approaches seemed to be less ideal for our model, as the feature selection picked a very small set of features, that actually gave higher generalization error. It seems as though such feature selection techniques undermine the built-in feature selection in our ensembles, and again could make our model vulnerable to overfitting and the curse of dimensionality. Instead we choose to remove only the obviously redundant features. We picked the best feature among each subset that had higher correlational coefficients than 0.95. After this selection there was in fact only 5 GLCM features left, namely GLCM-Contrast ($d = 12$), GLCM-ASM ($d = 4$), adaptive GLCM (4 vs. 1), adaptive GLCM (2 vs. 3), adaptive

GLCM (3 vs. 4). Removing these features did not affect the result of the cross-validation on the training-sets.

Table 8.3: Feature evaluation for the M51 dataset, with the accuracy for the individual features applied to the test-set. The CCR is presented in percentage. Feature importance is a measure of how large a share of the total reduction on the impurity measure, that each feature contributed to.

Features	CCR	(RF) Feature Importance	(GBT) Feature Importance
Max Diameter	89.06	0.0969	0.0373
Circularity	87.50	0.0913	0.0160
Hu Moment ϕ_1	84.38	0.1054	0.0317
Adaptive GLCM (1 vs. 4)	83.59	0.0665	0.1904
Area	80.47	0.0969	0.0355
Eccentricity	77.34	0.0878	0.2246
Perimeter	75.78	0.0901	0.0362
Bending Energy	68.75	0.0585	0.0263
Variance	68.75	0.0498	0.0261
GLCM-Contrast (d=12)	68.75	0.0507	0.0316
Concavity Depth	68.75	0.0049	0.0048
Mean Intensity	65.62	0.0191	0.0068
Convex Hull Deficiency	65.62	0.0220	0.0089
Adaptive GLCM (2 vs. 4)	60.16	0.0078	0.0054
Skewness	57.81	0.0061	0.0048
IOD	57.81	0.0204	0.0108
Jaggedness	57.81	0.0449	0.0069
Adaptive GLCM (2 vs. 3)	56.25	0.0170	0.0068
Summed Laplace	51.56	0.0307	0.0108
Entropy	48.44	0.0087	0.1457
Symmetry	43.75	0.0016	0.0102
Overlap	43.75	0.0022	0.0160
Blurred Edge	42.97	0.0017	0.0105
mean Fourier	42.97	0.0051	0.0035
GLCM-ASM (d=4)	39.06	0.0076	0.0360
Cut Cell	36.72	0.0015	0.0276
Distance to Ellipse	33.59	0.0008	0.0108
IOD Balance	33.59	0.0031	0.0082
Tennengrad Variance	32.81	0.0010	0.0034

In Table 8.3 we present an evaluation for each of the features, so we can easily compare their discriminatory power for cell-type classification. The CCRs is calculated for each of the features individually, and will therefore contain no information in regard to interaction effects or mutual information between the features. As we can see both circularity, maximum diameter, area and perimeter are ranked among the top scoring features. These features are highly correlated, so it is not necessarily a good choice to use them all in a classification. Each feature provides much information individually, but they may not contribute

Table 8.4: Feature correlations for the top 6 features in table 8.3.

	MD	C	HM	AG	A	E
(MD) Max Diameter	1.00	-0.91	0.92	0.77	0.87	0.75
(C) Circularity	-0.91	1.00	-0.90	-0.53	-0.61	-0.89
(HM) Hu Moment ϕ_1	0.92	-0.90	1.00	0.60	0.70	0.80
(AG)Adaptive GLCM (1 vs. 4)	0.77	-0.53	0.60	1.00	0.91	0.27
(A) Area	0.87	-0.61	0.70	0.91	1.00	0.39
(E) Eccentricity	0.75	-0.89	0.80	0.27	0.39	1.00

much additional information when combined with other features. We can see that *max diameter* can separate 89.06% of the samples in the M51 test set. This means that the other features are essentially included to separate the remaining 10% of the cells. This means that it is not necessarily the individual accuracy that characterize a good feature. If a feature can only separate 5% of the cells, it may still be valuable if those cells are among the 10% that *circularity* can not separate.

Feature importance for a tree is a measure of how much a feature contributes to the total decrease of the impurity function. The total decrease in impurity means the impurity of the initial un-split populating, minus the summed impurity of the samples in the leaf nodes. For each node where the feature is used, we calculate how large the reduction in impurity were, and sum these values. The final feature importance is then this summed reduction divided by the total reduction. When we use an ensemble of threes, as with gradient boosting and random forest, we simply take the average of the feature importance in each tree.

The measures of feature importance will contain both information about interactions and correlated features. If one feature is already chosen, it will be less likely that correlated features are chosen next. This is true for both methods of classification. Features that interact with a chosen feature will have a greater chance of being picked. For the gradient boosting algorithm with a tree depth of only 3, there is of course little room for interaction effects. Despite this, picking one feature will make some features more likely to get picked next, compared to others. The gradient boosting selects the features that are best for separating out those samples it could not separate in the previous iteration. A feature correlated with previously selected features will then be less likely to provide further separation.

A problem with the *features importance* measure is that its value may be somewhat arbitrary, especially for the gradient boosted tree. If one feature is picked early in the boosting procedure it will affect how all the other features are chosen. With a slightly different training-set a different feature may be picked first and the final result can potentially be completely different. In table 8.3, we can see that *eccentricity* and *Adaptive GLCM (1 vs. 4)* have especially high value of feature importance for the gradient boosted trees (GBT). This is typically an effect of being among the first selected features. As we know that the circularity is related to eccentricity, this may also explain why circularity, even with a CCR of 87.50%, contribute little to the gradient boosting classification. Table 8.4 shows the correlation coefficients for the top 6 features. We can

se that almost all are highly correlated. This type of correlation is obvious for many of the features. We know that eccentricity, circularity, ϕ_1 and max diameter describe many of the same things. Generally the GLCM feature should be completely unrelated to these features, but as we can see, they still share some of the variation. This is obviously because they are all related through class label. Adaptive GLCM is related to area because small cells tend to be dark, have little patterns and high contrast, compared to larger cells. To find uncorrelated features, we would have to measure features that are completely unrelated even in the biology, and they are hard to find. The strength of our classification models is that we do not need to find such features, but can quite safely extract information, even from highly correlated features.

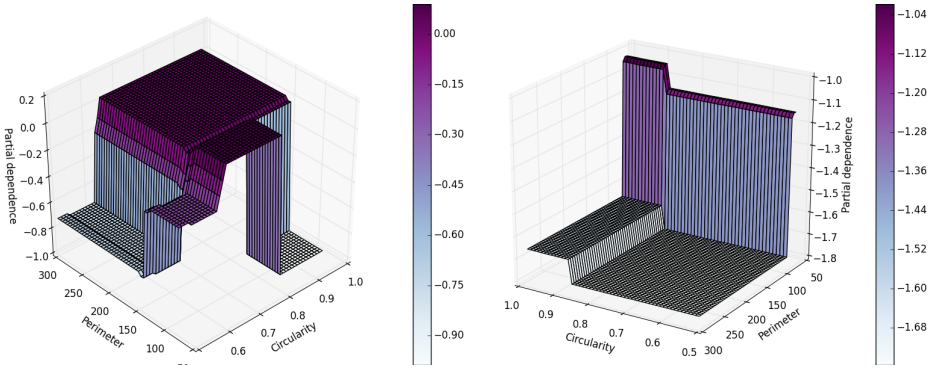
For random forest classification, a feature picked at a root node of a tree will affect all the choices down to the bottom of the tree, and the initial feature may be quite arbitrarily chosen, but this selection does not have an effect on the feature selection in other trees. This means that the effect of the initial selections will be averaged out across the number of trees. Since the random forest algorithm will have to pick one of the features in a small subset, the feature importance can also get more distributed among all of the features. If all the available features were quite irrelevant, at least one of the irrelevant features would be selected, and gain in feature importance.

It is interesting to note that the best features, especially in terms of features importance, are either morphological or textural features. *Variance* have a relatively high CCR compared the importance. One reason for this may be that the textural features often also provide various information regarding the gray-level histogram and definitively some information about the variance. Entropy on the other hand contribute greatly to the gradient boosting, despite a relatively low CCR. It may be that it contribute information that is unique compared to the other important features, separating subgroups that may otherwise be strongly interleaved.

Partial Dependence

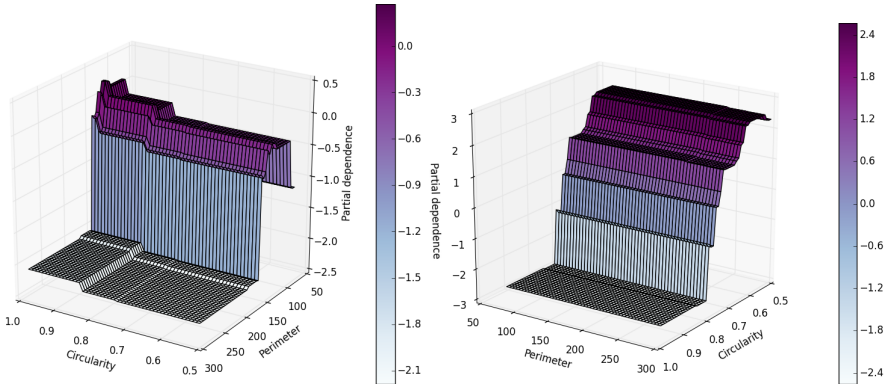
We can use partial dependence to investigate how the different features influence the classification and how the features interact to provide more information. In figure 8.1 we can see that a cell is more likely to be classified as K1 if it has a high value for *circularity* and a long perimeter. This is natural, as K1 and K4 can have similar perimeter length, but then K4 are almost always less circular. Most cells with short perimeter are K2 or K3 cells, but they tend to be very circular in shape. Cells are far from circular, but have a short perimeter tend to be K1 cells. K4 tends to have a longer perimeter than K2 and K3, but separating those are not much of a problem. Since you cannot separate K1 and K4 by perimeter, the decision boundary created for K4 is therefore unaffected by the perimeter length, but focus primarily on circularity.

Figure 8.3 shows the effect of the adaptive GLCM, where we summed the regions where K1 had higher GLCM values than K4. Naturally we see that cells with a high value on this feature are more likely to be classified as K1. What comes as a small surprise is that the K4 decision boundary is so unaffected by this feature. It shows a small trend that cells with a low value on this feature are more likely to be classified as K4, but this trend is dwarfed by the huge effect of *circularity*. The reason for this may be that K2 and K3 tend to have a small



(a) K1

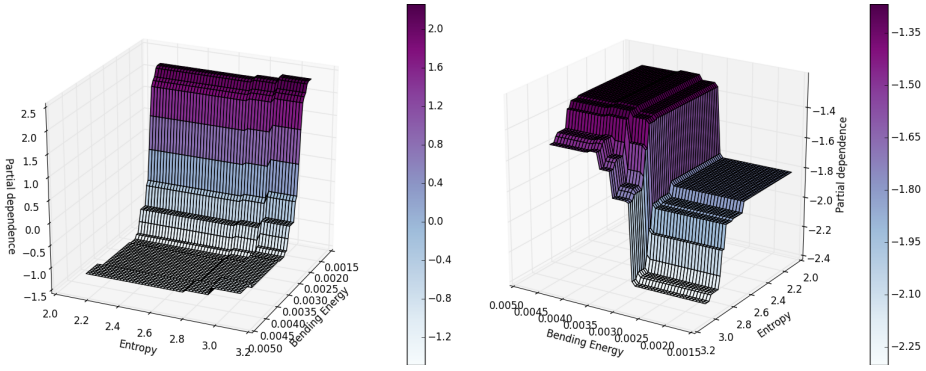
(b) K2



(c) K3

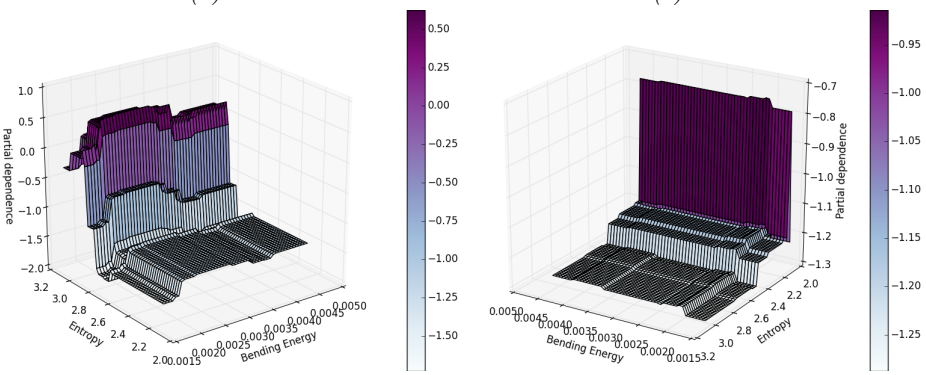
(d) K4

Figure 8.1: A partial dependence plot, displaying the average effects of perimeter and circularity. The boundary was created by the gradient boosting algorithm. On the right there is a color bar indicating how significant the effects are. We can se that there is quite a dramatic effect of circularity on K4.



(a) K1

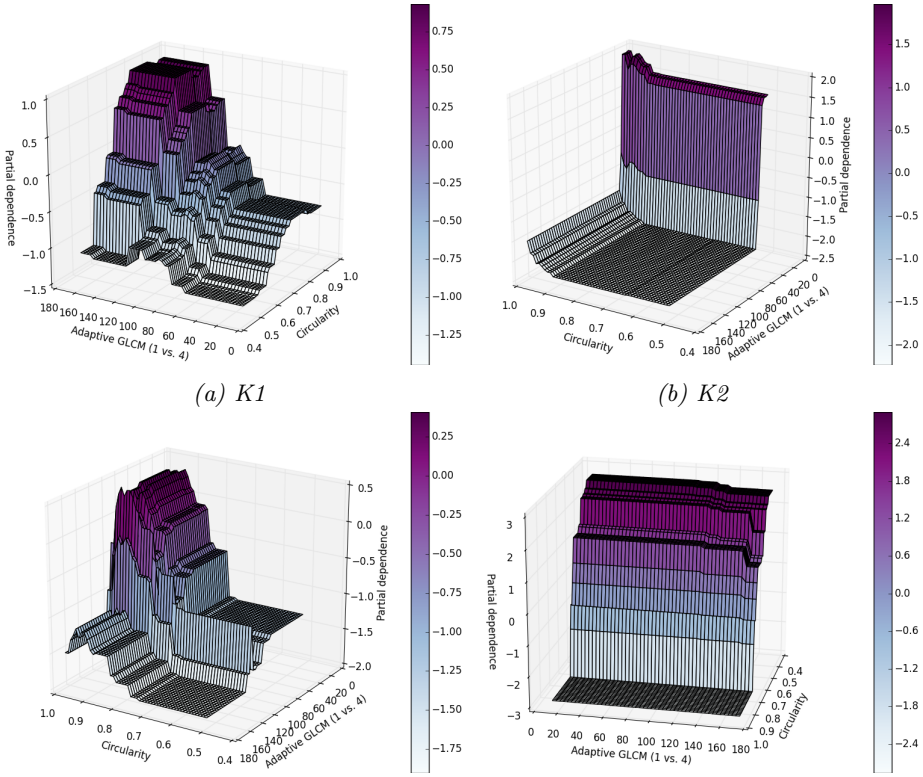
(b) K2



(c) K3

(d) K4

Figure 8.2: A partial dependence plot, displaying the average effects of bending energy and entropy. The boundary was created by the gradient boosting algorithm. On the right there is a color bar indicating how significant the effects are.



(a) K1 *(b) K2*

(c) K3 *(d) K4*

Figure 8.3: A partial dependence plot, displaying the average effects of Adaptive GLCM and circularity. The boundary was created by the gradient boosting algorithm. On the right there is a color bar indicating how significant the effects are.

(a) K4 *(b) K4* *(c) K1*

Figure 8.4: Here we have a set of oblong cells taken from M51. We can see the K4 cells have two pointed ends, and therefore a higher bending energy.

value for this feature as well, so it is not really a good feature for separating K4 alone. For K2 and K3 this adaptive GLCM feature has a much larger effect. If we look closer at the magnitude of the Mahalanobis distances in figure 6.17, we can in fact see that they are much larger between the other classes, compared to K4, and that they have differences in similar regions as the $J(K1, K4)$ matrix.

From figure 8.2 we can see the effect of the *bending energy*. It proves valuable in the separation of K1 and K4. As bending energy is the sum of the squared curvature, K4 will get a high value relative to its size, as it has very high curvature at the pointed ends of its oblong shape. There have also been some indications that even oblong K1 cells tend to have less pointed ends. From figure 8.4, we can see some cells that are typically separated by their *bending energy*.

8.1.2 Classification Results

In this section we will present both the results for cross-validation of the training data, and for training on the training-sets and testing on the independent test-sets. Presenting the classification of the training-sets as well as the test-sets can give us insight into how the feature generation procedure has affected the results. This also gives us an opportunity to test the algorithm on a much larger set of data. Since the distribution of cells among the classes is very skewed, we are faced with a dilemma. If we use the skewed data-set it will have strange effects on our results. With the large number of K1 cells, the accuracy will basically only tell us how good the algorithm is to classify K1. We could in reality have a classifier that placed all cells in K1 and still got a decent CCR. Instead we decided to use even training- and test-sets. This on the other hand have a drastic effect on the amount of data, available. We cut the number of cells in each class down to the number of the least frequent class. In some cases that leaves us with only 2 % of the original data, for testing purposes. Therefore we run the classification multiple times, where we change the samples from the most frequent classes each time. In that way we can use all the test samples. We then report the average confusion matrix and CCR, but the most numerous classes are averaged from a large number of samples, while the more rare cells used only once. This means that we may not have a representative evaluation for these classes. Doing cross-validation on the training-set will therefore give

us an opportunity of testing the algorithm on a larger data-set. Even if the results can be slightly optimistic we can get an impression of whether these results match with the testing results.

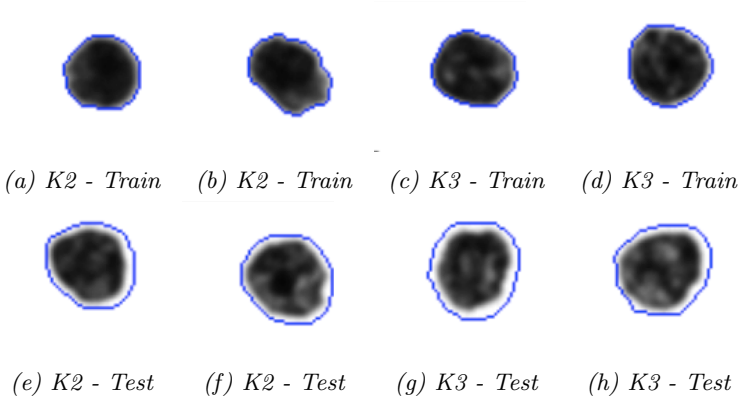
Figure 8.5: The leftmost cell is a K1 cells from the M51 test set, that ended up classified as K4. Second from the left we have a K4 cell correctly classified. They are not all to similar, but the leftmost cell actually look more similar to a regular K4 cell than the second cell. The third cell from the left are a K1 cell classified as K3, and the rightmost cell is a correctly classified K3 cell.

The Table 8.5 present the results for the training-sets, while Table 8.6 give the results for the test-sets. M51 seems yield very good results and when investigating the errors we find that the cells confused between K1, K3 and K4 are generally very difficult do distinguish even visually. Both for M51 and L41 the results decreased on the testing sets compared to the training-sets. For M51 this seems quite natural as this was the set we used most in the search for features in chapter 6. L41 on the other hand has not been much used, so overfitting due to feature generation seems unlikely. With a small number of cells the poor results for the L41 test-set could be a coincidence, but this is of course a matter of concern when regarding the robustness of the classification. When investigating the L41 test- and training-set we found that they differed in some basic ways. We tested the Mahalanobis distance for each feature between the training and test-set. We found that *perimeter* and *area* which had large feature importance in the classification also had the greatest Mahalanobis distance. The difference was largest for the K2 class, where the distance was 1.86 for perimeter and 1.96 for area. With those two features removed we gained almost 2% in CCR. For K3 the *mean intensity* and *variance* had a distance of about 0.5.

For the PLM13 set we had a smaller training-set, but a larger test-set. There we found similar results for both the test and training. This set was not used much in the development of features, and these similar results come as no surprise. The test- and training-set of PLM13 had patient number from 500 to 520 and 521 to 530. Therefore the train and test-set may have been edited closer in time, and be more similar as a result of a more similar segmentation algorithm or being classified by the same experts. With a larger test-set we may also even out the differences in error rates. What we do not know is whether some diagnostic groups are more prone to have cells misclassified than other. This should be investigated as this could affect further results, but such an investigation is hard in our position where we do not have any knowledge of these prognostic groups. As we can see from table 8.5 and table 8.7, the primary difficulty was the separation of K2 and K3. It turns out that some of the patients have separate K2 and K3 cells very clear cut, at about 1000, similar to what we found in figure 8.7. As area usually is the best way of separating K2 and K3 we can easily understand why these cells are misclassified.

The result for the inter-observer set on the other hand gave somewhat more

disappointing results. With such a small number of samples this could again be somewhat due to chance, but we found significant differences between the training and test-set also here. We found that the inter-observer set from one patient belonged to the minority group of patients where K2 and K3 are separated at an area of 1000. With further investigation we found that not only did the area differ, but also the color and texture. To be more exact, in the training-set the K2 cells were usually very dark, almost completely black. In the inter-observability set on the other hand they had more of a pattern, dark with white speckles, also somewhat larger and looked more like K3 cells in general. For K2 the Mahalanobis distance was 4.82 for *Mean Intensity*, 2.2 for *gray-level variance* and 2.1 for *gray-level entropy* between the classes. The differences may in part be explained by differences in segmentation, as we found the cells in the inter-observer set to be slightly more over-segmented than in the training-set. In figure 8.6 you can see some typical examples of K2 samples from the training and test data. Over-segmentation obviously affects both intensity, variance, entropy and area. Another possible explanation for these differences may be found in the classification of the cells. From figure 8.7 it can look like the cells are simply classified to K2 or K3 based on whether they had an area larger than 1000. This awakes some suspicion that perhaps different presorting algorithms have been used. It certainly seems unlikely that the type of separation of cells found in figure 8.7 can be made by pure coincidence. It could also be that the imaging techniques have improved and that we now see texture in what earlier just looked black. The third possibility is obviously that we did not have a representative training-set, and that we have not captured that natural variation of K2 and K3 cells.



(a) K2 - Train *(b) K2 - Train* *(c) K3 - Train* *(d) K3 - Train*

(e) K2 - Test *(f) K2 - Test* *(g) K3 - Test* *(h) K3 - Test*

Figure 8.6: The top row are cells from the PLM13 training-set, while the bottom row are cells from the PLM13 inter-observer set. The cells are all found along the boundary of K2 and K3, in the respective dataset. The two left most columns are K2 cells and the right most columns are K3 cells. We can see how the K2 cells from the training-set are very dark, while the cells from the test-set have some lighter texture and generally look more like K3 cells. In this figure 1 cm correspond to 4.98 μm. The biggest effect seem to be due to over-segmentation, which is quite apparent when comparing the two rows. This may explain the difference in cell size.

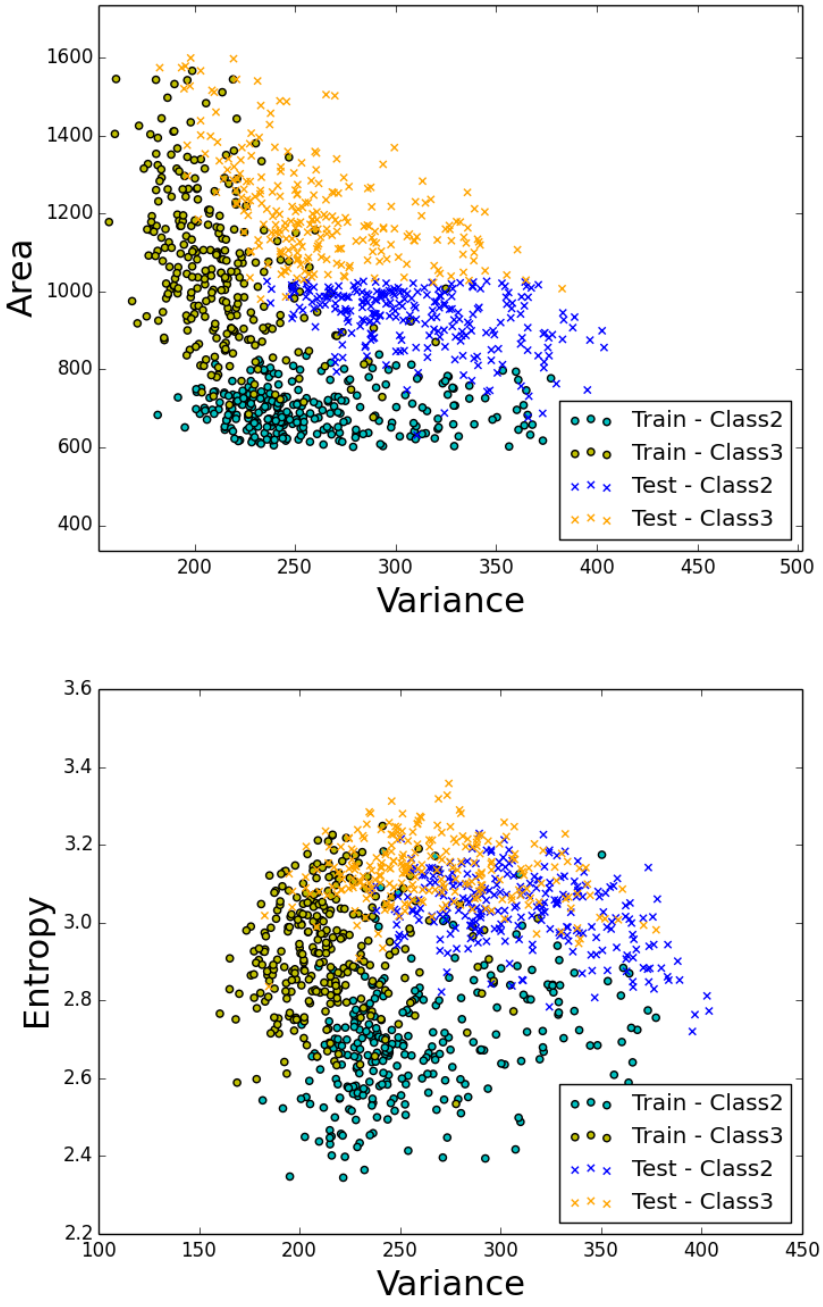


Figure 8.7: The scatter plots show how the PLM13 training-set and the PLM13 inter-observer set differ in terms of variance, entropy and area. The circular markings represent samples from the training-set, while the crosses represent the inter-observer set.

It is clear that there are differences between the training and test-set for both the L41 and PLM13 set. In other words it seems like our algorithm is not robust agains large changes in gray-level, variance and scale. The question is then how we could have avoided this problem or if it should be avoided at all. It is clear that we want a general algorithm that can withstand some changes in the data material and be as robust as possible. The problem is that our approach has to use some differences to classify the cells, in other words we cannot make an algorithm that is invariant to all types of changes. We have to somehow look for what is true differences between classes and what differences are there by pure coincidence. This problem could have been avoided either if we had a priori information that size, gray-level variance and color were unreliable features or if we had a large representative training-set, so the classifier would learn that these features were not as strong as they seem to be. With a larger more heterogeneous training-set, or less features we could end up with a less separable dataset, but one often have to make tradeoffs in order to achieve higher generalizability. In our case we had no a priori information and based all or knowledge on the training data ourself, so we believed the training data to be representative in the same way as the classifier did.

We can note that in many of the tests we achieved close to the expected results of human experts. We also experienced that the results are less robust to noise than what we would hope. Based on the approach we used it is not surprising that the model is not robust to such changes, as we use quite few features in a very direct way. A possible way to improve the generalizability could be to use a normalization procedure, where the features depend in some degree on some average values in a dataset.

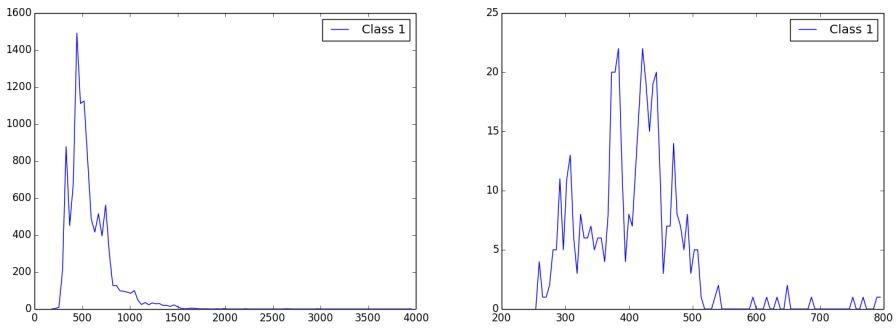


Figure 8.8: To the left we have the distribution of IOD for the correctly classified K1 cells and to the right we have the IOD distribution for misclassified K1 cells. We can see that the correctly classified cells have a much higher range of IOD.

We found that generally more low IOD K1 cells were misclassified compared to high IOD cells, as can be seen in figure 8.8. We found no difference in this distribution if we removed IOD as a feature, so leaving these features out is no cure to the problem. We still believe that since the K1 class is so frequent and the misclassified cells seems to be so similar to K3 or K4 that we can not tell the difference, we do not think that the bias introduced will have any effect on the further analysis.

Table 8.5: Classification results for cross-validation of the training-sets. On the left side of the table there is a confusion matrix, with the expert labels as rows, and the classification result as columns. The total number of cells classified are written inside the parenthesis after the name of the set.

Random Forest: M51-Training (2452) *CCR: 98.41%*

Class	K1	K2	K3	K4	Precision	Recall
K1	589	0	3	21	0.97	0.96
K2	0	613	0	0	1.00	1.00
K3	6	0	607	0	0.97	0.99
K4	9	0	0	604	0.97	0.99

Gradient Boost: M51-Training (2452) *CCR: 98.36%*

Class	K1	K2	K3	K4	Precision	Recall
K1	587	0	7	19	0.98	0.96
K2	0	613	0	0	1.00	1.00
K3	4	0	609	0	0.99	0.99
K4	10	0	0	603	0.97	0.98

Random Forest: L41-Training (2084) *CCR: 97.69%*

Class	K1	K2	K3	K4	Precision	Recall
K1	488	0	10	23	0.98	0.94
K2	0	521	0	0	0.98	1.00
K3	4	0	507	0	0.98	0.97
K4	5	0	1	515	0.96	0.99

Gradient Boost: L41-Training (2084) *CCR: 97.45%*

Class	K1	K2	K3	K4	Precision	Recall
K1	495	0	6	20	0.97	0.95
K2	0	613	0	0	0.99	1.00
K3	9	5	520	1	0.99	0.97
K4	6	0	0	515	0.96	0.99

Random Forest: PLM13-Training (1288) *CCR: 95.26%*

Class	K1	K2	K3	K4	Precision	Recall
K1	316	0	3	3	1.00	0.98
K2	0	296	26	0	0.91	0.92
K3	1	28	293	0	0.81	0.91
K4	0	0	0	322	0.99	1.00

Gradient Boost: PLM13-Training (1288) *CCR: 94.18%*

Class	K1	K2	K3	K4	Precision	Recall
K1	312	0	5	5	1.00	0.97
K2	0	281	41	0	0.92	0.87
K3	0	23	299	1	0.87	0.93
K4	1	0	0	321	0.98	1.00

Table 8.6: Classification results for the independent test-sets, of the two "old datasets". On the left side of the table there is a confusion matrix, with the expert labels as rows, and the classification result as columns. The total number of cells classified are written inside the parenthesis after the name of the set.

Random Forest: M51-Test (128) *CCR: 96.64%*

Class	K1	K2	K3	K4	Precision	Recall
K1	28.2	0.0	0.5	3.3	0.98	0.88
K2	0.0	32.0	0.0	0.0	1.00	1.00
K3	0.5	0.0	31.5	0.0	0.98	0.98
K4	0.0	0.0	0.0	32.0	0.91	1.00

Gradient Boost: M51-Test (128) *CCR: 95.94%*

Class	K1	K2	K3	K4	Precision	Recall
K1	27.1	0.0	0.5	4.4	0.99	0.85
K2	0.0	32.0	0.0	0.0	1.00	1.00
K3	0.3	0.0	31.7	0.0	0.98	0.99
K4	0.0	0.0	0.0	32.0	0.88	1.00

Random Forest: L41-Test (152) *CCR: 92.24%*

Class	K1	K2	K3	K4	Precision	Recall
K1	31.2	0.0	5.2	1.6	1.00	0.82
K2	0.0	38.0	0.0	0.0	0.88	1.00
K3	0.0	5.0	33.0	0.0	0.86	0.87
K4	0.0	0.0	0.0	38.0	0.96	1.00

Gradient Boost: L41-Test (152) *CCR: 91.78%*

Class	K1	K2	K3	K4	Precision	Recall
K1	32.2	0.0	4.0	0.8	0.95	0.85
K2	0.0	38.0	0.0	0.0	0.88	1.00
K3	0.6	5.0	32.4	0.0	0.89	0.85
K4	1.0	0.0	0.0	37.0	0.95	0.97

Table 8.7: Classification results for the independent test-sets, of the two "new datasets". On the left side of the table there is a confusion matrix, with the expert labels as rows, and the classification result as columns. The total number of cells classified are written inside the parenthesis after the name of the set.

Random Forest: PLM13-Test (652) *CCR: 96.21%*

Class	K1	K2	K3	K4	Precision	Recall
K1	157.8	0.0	1.2	4.0	1.00	0.97
K2	0.0	156.0	7.0	0.0	0.93	0.96
K3	0.0	12.5	150.5	0.0	0.95	0.92
K4	0.0	0.0	0.0	163.0	0.98	1.00

Gradient Boost: PLM13-Test (652) *CCR: 95.49%*

Class	K1	K2	K3	K4	Precision	Recall
K1	159.3	0.0	0.5	3.2	1.00	0.98
K2	0.0	151.0	12.0	0.0	0.92	0.93
K3	0.0	13.2	149.8	0.0	0.92	0.92
K4	0.5	0.0	0.0	162.5	0.98	1.00

Random Forest: P13 - Inter-Observer (44) *CCR: 92.23%*

Class	K1	K2	K3	K4	Precision	Recall
K1	10.7	0.0	0.0	0.3	1.00	0.97
K2	0.0	10.4	0.6	0.0	0.87	0.95
K3	0.0	1.6	9.4	0.0	0.94	0.85
K4	0.0	0.0	0.0	11.0	0.97	1.00

Gradient Boost: P13 - Inter-Observer (44) *CCR: 91.81%*

Class	K1	K2	K3	K4	Precision	Recall
K1	10.5	0.0	0.0	0.5	0.91	0.95
K2	0.0	11.0	0.0	0.0	0.84	1.00
K3	0.0	2.1	8.9	0.0	1.00	0.81
K4	1.0	0.0	0.0	10.0	0.95	0.91

The overall difference between the two classification algorithms was small, but for our application random forest generally outperformed gradient boost. The average accuracy for the *random forest* was 95.53% and 95.0% for the *gradient boosted trees*. It may be that gradient boost could have been improved with a more thorough grid-search for meta-parameters, as gradient boosted trees have been found to be more sensitive to parameter selection [11]. Still with a classification model that works well with different meta-parameters may be the best choice for our application, as the same model may work better for many different datasets.

As there was only small differences between the classification models, this type of study, with a very complex dataset is unsuited for a conclusive model evaluation. To get a true picture of the differences between the models we would have to perform a much more rigorous study, where we for example manipulate the labels, to investigate the robustness against mislabeling. We found this type of investigation to be outside the main focus of this study and it was therefore not performed. With this result we decided to only report the result of the *random forest* in next classifications, but on the data we did test our results was very similar for both models. With similar results, we find that a random forest is the best model for our application, as it seems to be more robust to changes its meta-parameters and because it is much faster, are easily parallelized and achieve slightly better performance.

8.2 Feature Value Thresholding

Throughout the development of the features for removing K6, we have been familiarized with the training-sets. Our features for filtering out K6 cells were primarily developed using the P02 and PLM14 dataset, but we also did some investigation on M51. We may have over adapted the features to account for the variation in these sets. To gain a better understanding of the discriminatory power of the features, we will attempt thresholding on the independent test-sets. For evaluation we first set a threshold based on the training samples, and then use the same threshold for the test data.

As the goal of thresholding is not to obtain the best possible CCR, but rather to avoid loosing any important K1 cells, the threshold is set through visual examination of the cells. The threshold is set to a level where only cells that could have been classified as K6 are removed. All cells with feature a value above a certain threshold will be detected as K6. We will report how many cells from other classes than K6 that is lost, and how many K6 cells we can remove with this feature. To evaluate whether the features work as intended, a visual inspection is performed.

8.2.1 Overlapping Cells

For the M51 training-set, 500 was found to be a reasonable threshold for the *overlapping* value. The main problem for setting the threshold was the many under-segmented K1 cells in M51. Over-segmented cells have deep concavities that increase the general overlapping value. Some loss of the most over-segmented K1 and K4 cells had to be accepted to be able to set a threshold without modifying the feature calculation. A threshold of 500 removed 1.43 %

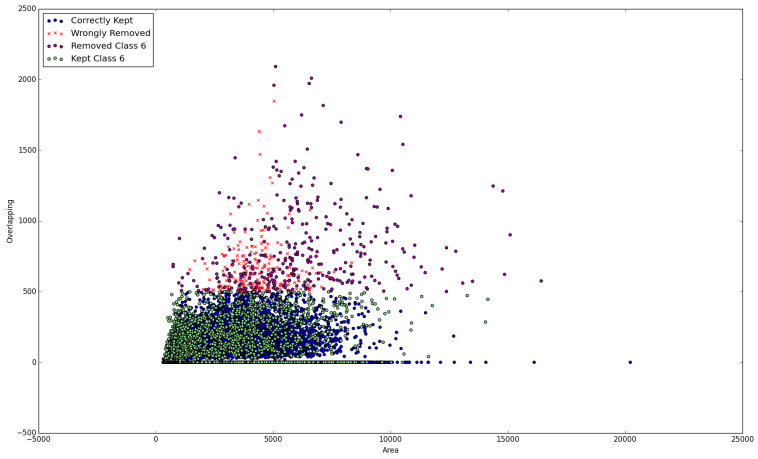


Figure 8.9: Plot of thresholding on the overlapping *feature value at 500, for the M51 test-set. There are quite a few K1 removed, but they are at least boundary case overlapping or have deep under-segmentation.*

of the non K6 cells and 10.42% of the K6 cells. We found no K1 cells removed that could not be interpreted as overlapping or very under-segmented. We also managed to remove most of the obviously overlapping cells.

Table 8.8: The percentage of cells removed from each class when thresholding the overlapping value on 500.

	M51-Train	M51-Test
Class 1	1.8 %	1.04 %
Class 2	0.0 %	0.0 %
Class 3	0.09 %	0.0 %
Class 4	2.72 %	3.92 %
Class 6	10.42 %	5.51 %
Non K6	1.43 %	0.91 %

A threshold for the M51 test-set removed 0.91% of the non K6 and 5.51% of the K6 cells. The figure 8.9 illustrate the cut in the distribution, plotted with *area* on the x-axis and *overlapping* on the y-axis. The K1 cells misclassified as K6 were not all clearly overlapping, but had some artifacts. In figure 8.10 we give an example of cells typically wrongly removed by a thresholding in terms of overlapping. One of the main issues with M51 is when deep concavities align with a very dark pattern in the cell. We tested a logarithmic transform on the edge image, that we use to find and evaluate the *overlapping* features, but found that these patterns were so strong, that they usually still dominated the edge image. For some cells with a grainy texture the logarithmic transform also made the real edges weaker compared to the normal texture of the cell, so we found a very short path across the cell, instead of following the best edge.

Table 8.9: The percentage of cells removed from each class when thresholding the overlapping value on 500.

	L41-Train	L41-Test
Class 1	0.98 %	1.14 %
Class 2	0.0 %	0.0 %
Class 3	0.3 %	0.3 %
Class 4	2.88 %	3.45 %
Class 6	5.49 %	6.03 %
Non K6	0.91 %	1.06 %

For the L41 training-set we used a threshold of 380. This removed 1% of the non K6 and 4.5% of the K6 cells, and most of these cells were clearly overlapping. For the test-set 1.5% of the non K6 and 6.4% of the K6 cells were removed. Even though we did not remove a large part of the cells it seems that for the test-set this threshold should have been set higher, as some cells, obviously not overlapping were filtered away. For the training-set a threshold of about 450 seems to be better.

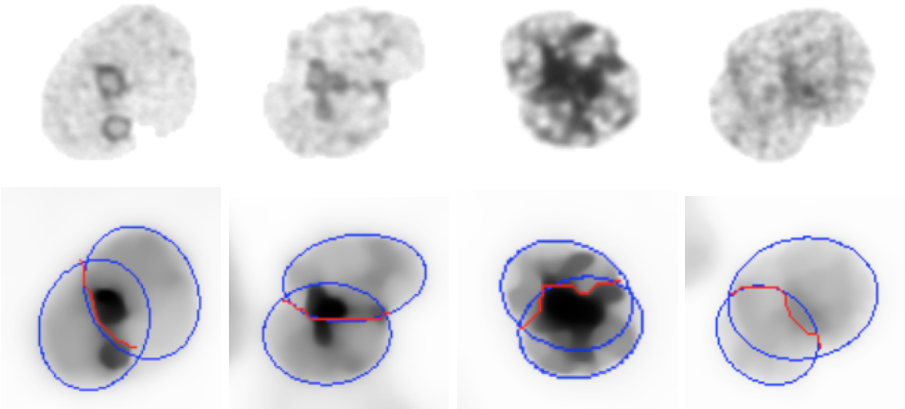


Figure 8.10: Here we present 3 K1 cells, removed from the set due to a overlap *value higher than 500. The two first cells to the left are very typical for the removed K1 cells, in that they have deep concavities and some dark patterns matching with concavity. We are not sure whether the third cell from left is overlapping or not, but it was labelled as K1. This cell illustrates a typical problem with the algorithm, were the path adapts to much to the underlying texture, and in our opinion get a to high value relative to how "overlapping it looks". The rightmost cell is presented as it illustrates a more general problem in finding overlapping cells. It is very hard to determine whether this cell is overlapping or not, and therefore hard to create general rules for classification.*

All cells from the PLM13 data-set with a value over 400, could in our opinion been classified as K6. With this threshold we could remove 4.92 % K6 cells, with a loss of of only 0.69% of the non-K6 cells. Some very long K4 cells proved problematic, as bent cells often have deep concavities. If these concavities

Table 8.10: The percentage of cells removed from each class when thresholding the overlapping value on 500.

	PLM14-Train	P02-Train
Class 1	0.68 %	0.65 %
Class 2	0.0 %	0.0 %
Class 3	0.26 %	0.0 %
Class 4	3.56 %	3.36 %
Class 6	4.92 %	5.94 %
Non K6	0.69 %	0.66 %

aligned with pattern inside that cell, they could get a high overlapping value. The PLM13 cells had generally a lower value of this overlapping measure, having more smooth edges and less concavities. Even with this threshold we were far from removing all the overlapping cells. To remove most of the overlapping cells we would need a threshold of about 320. This would also lead to a removal of several non-overlapping K1 and K4 cells, similar to the cells in figure 8.11. If we had set the threshold to 320, we would removed 5.73% K6 cells and still only 0.86% of the non-K6 cells. With less rough edges it is perhaps harder to find a reason for why a cell could be classified as K6. This fact may have lead to a lower tolerance for loosing non-K6 cells for this data-set.

For the PLM13 test-set we found that the threshold of 400, removed 5.94% of the K6 cells and 0.66% of the non-K6 cells. We found no obvious mistakes, but similar to the training-set we found that there were many undetected overlapping cells. In figure 8.13, we have presented some of the cells that were detected, and in figure 8.12 we present some cells that remained undetected.

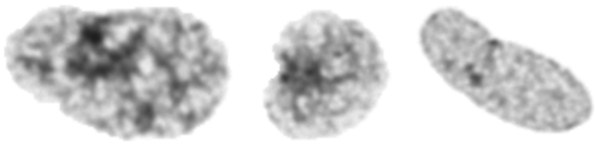


Figure 8.11: Three K1 cells from PLM13, with a value for overlapping *just below the threshold of 400.*

Overview

In all datasets some part of K4 was removed. This was both because the bent cells had high convexity, but because a dark spot on a very thin cell cover a much larger part of the path across the cell. The mean edge will therefore be comparably large.

As a general observation we found detection of overlapping cells incredibly hard. Cells that may seem obviously overlapping for a human eye, can be very hard to detect with a general set of rules. The leftmost cell in 8.12, is a typical example of this, with very small concavities, and irregular and incoherent edges inside the cells. One of the main problems is the heterogeneity of the cancer cells, making it hard to predict the possible shapes and variations. We cannot

trust the cell mask alone, as many cells look clearly overlapping judging by the cell mask, but as long as there are no effect of the overlapping in the cell texture, the cell rarely classified as overlapping. Minute changes in the texture can be detected by humans, but when we do not know how the textures can change in advance it is very hard to write an algorithm that detects them in general. We did experiments with using texture feature images, to base our *cost image* on. Here we calculated different GLCM-features in a sliding window, and the pixels in our texture image consisted of the GLCM feature for the surrounding area of each pixel. The problem was that in order to detect the differences in texture between two overlapping cells, we had to use many different GLCM features. With many different texture images the probability that an edge in one of the images match our path by pure coincidence, became very high. The attempt of utilizing the texture was therefore rejected quite early in our process. We still acknowledge that in order to get a perfect detection we may need to incorporate the texture information in some way.

More directly related to our current algorithm, it is clearly an issue that we have no restrictions on our path in terms of smoothness. For example for the third cell from the left in in figure 8.10, the path found is very unlikely, as for most samples the overlapping part of the cell has similar smoothness as the rest of the cell. Allowing for such paths give some cells a far to high value on the overlapping measure, compared to how overlapping they look visually. A solution to this could be to add some constraint to the path, similar to the stiffness matrix for a snakes in an active contour model [10]. We can imagine a solution where the stiffness matrix of a snake is determined by the different parts of the cell contour, but this is not an easy task. A more easily applicable solution would be to incorporate the bending energy of the path somehow. As always we have to find a balance, as this could add to an already existing problem. The algorithm tend to choose the shortest path instead of a path with the strongest edges. We provided some examples where the shortest path was chosen in favor of the path with the strongest edge in figure 8.12. By also including the bending energy of the path, the algorithm would favor short straight paths even more. We have tried to make the cost of path-length the mean value of the cost image in stead of a sum, and also to scale the cost image between 1 and -1 instead of 1 and 0. This lead to the algorithm finding strange and long paths through the cell, to fit black spots perfectly. It is still possible that a combination of this and an added cost of increased bending energy would be a good approach, but we did unfortunately not have time to implement this for our project.

8.2.2 Cut cells

On the P02 and PLM14 sets that we primarily developed this feature for, we found quite promising results, unfortunately we did not get the same results when we investigated the test-sets. The result for these two sets can be found in table 8.11. We found that one of the main reasons for this actually was that PLM14 contained a lot of cut and damaged nuclei, compared to the other data sets. The results from the other test-sets are not as grim as they may seem, as many of the "misclassified" cells are actually clearly cut. This was exactly the reason for developing these features.

To remove all cells in the M51 training-set that we interpret as cut, we would have to set a threshold on our *cut measure* to about 430, but then we would also

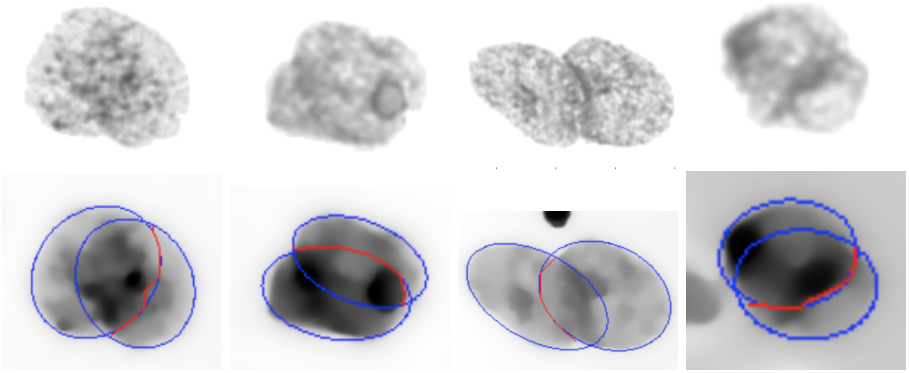


Figure 8.12: This is a set of overlapping cells that was not found by our approach, but they all had values close to the threshold. The tree left most cells, especially seems to illustrate the problem where the shortest path is pick instead of the one with the best edges, and therefore the cell end up with a lower overlapping value then it should, based on a visual inspection.

Table 8.11: Thresholding on the cut cell feature value on 580, for both the PLM14 and P02 training-sets.

	PLM14-Train	P02-Train
Class 1	3.63 %	5.32%
Class 2	9.09 %	0.0 %
Class 3	6.75%	4.84 %
Class 4	1.52 %	0.0 %
Class 6	43.81 %	29.04 %
Non K6	3.77 %	5.21 %

need to sacrifice several K1 cell that were not cut, and should be kept in the data-set. With a threshold of 430 we lose 3.7% of the non K6 cells, and remove 22.5% of the cells in K6. To avoid loosing perfectly good cells we actually need to set the threshold up to 520. Then we lose 1.5% of the non K6 cells and remove 13.5% of the K6 cells, as presented in table 8.12. The main problem for this measure is that cells with very rough edges due to over- or under-segmentation can get very high curvatures. The size of the over-segmented areas does not depend much on the cell size. The effect of the over-segmentation is therefore larger for K2 and K3 cells, as they are relatively small. This effect is further reinforced by the fact that small cells have generally higher curvature. Finding a line from an over-segmented "spike" in the contour, to another high curvature point on the cell, is therefore easy with these cells.

For PLM13 the main problem is not over-segmentaion, but instead K2 and K3 cells with a uneven edges or triangular shapes. In table 8.13 we can see how these cell affect the result. They are visually indistinguishable from a huge part of the K6 set, so there seems to be no consensus among the experts on this issue. We do loose many of these K2 and K3 cells, but an even larger group of indistinguishable cells in K6 are correctly removed. Even though these cells are

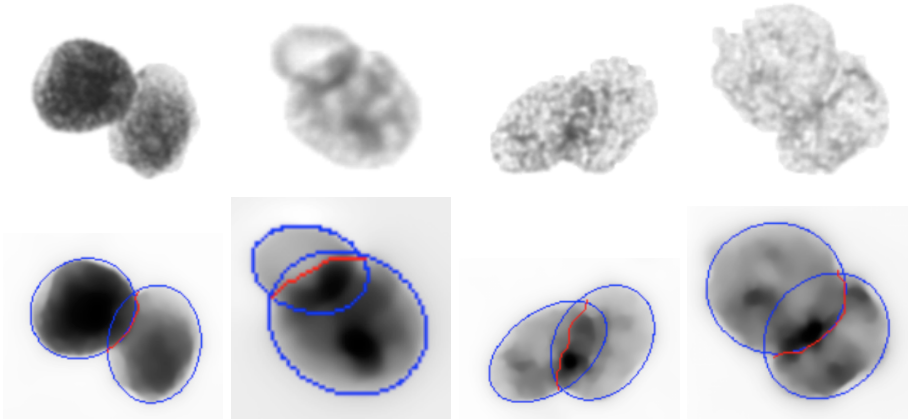


Figure 8.13: These cells were detected as overlapping. The second row display the filters images used to find the overlap. The ellipses found are marked in blue, and the path is marked in red. The two cells to the left are quite obviously overlapping and easily found by our approach. The third cell is also visually clearly overlapping, but it does not look like the algorithm found the best path. This again illustrates one of the problems with the algorithm, namely that it tends to take the shortest path, even when another paths seems more suitable. The right most cells, is found to be overlapping and certainly looks that way, but we precent this cell since similar cells are often not labelled as overlapping.

K6 cells or could easily be classified as K6, we have to note that our measure does not achieve its goal. Our attempt was to create a feature that specifically removed cut cells, while the cells in figure 8.14 are definitively not cut.

For the K1 cells the story is different. We looked through 100 randomly selected cells over the threshold. Most of the cells were clearly cut, and all were either questionably cut or over-segmented. In figure 8.15, we provide typical examples of these three groups.

For the PLM13 test-set there result were much better in terms "misclassification" of all the other cells. What we actually found were that the test-set had much less cut cells in K1, and the difficult cells like the ones presented in figure 8.14 were typically labelled K6. Even the K1-K4 cells that were thresholded out in this attempt were mostly cut or over-segmented. The different class distributions for the PLM13 test-set on the *cut cell* measure can be found in figure 8.16.

For L41 we found that many of its cells would have been classified as cut in other sets. If we set the threshold to 520 as in M51, we can remove most of the obviously cut cells, but we also lose quite a few K1 and K3 cells that looks nearly indistinguishable from the cut cells. The results of this thresholding can be found in table 8.14. With our threshold we also lost some close to triangular K3 cells, that do not look cut. Since the K1 and K3 cells that looks cut often have very high value on the *cut* feature, setting the threshold higher will not give much of an improvement. Perhaps the main problem related to L41 on this feature is not directly over- or under-segmentation, but that they often are segmented in such a way that they get straight edges and sharp corners, looking

Table 8.12: The percentage of cells removed from each class when thresholding the cut cell value on 520.

	M51-Train	M51-Test
Class 1	1.47 %	2.23 %
Class 2	8.15 %	18.75 %
Class 3	8.15 %	10.79%
Class 4	2.49 %	2.94 %
Class 6	13.48 %	23.96 %
Non K6	3.16 %	3.57 %

Figure 8.14: The top row is K3 cells with cut *value over the threshold. The bottom row is cells found in K6 that looks very similar. All cells are from the PLM13 training-set.*

very similar to cut cells.

Overview

We see that the performance of this feature depends on the different segmentation algorithms used. For the old sets the problems were mostly a rough segmentation. For the newer sets on the other hand, there are problems related to artifacts in the segmentation algorithm, causing very high curvature on an otherwise smooth contour. As in most cases we also have problems with labels that appear unreliable. More directly related to the algorithm, we have problems with the triangular cells like the K3 cell, left most in figure 8.18 and how we can distinguish them from the K6 cell, second from the right. We might be able to adapt to some of these problem by adjusting the σ for the smoothing in the curvature measure, but it is hard to find one parameter that works well for all sizes. An alternative could obviously be to fit a function for σ to the cell area, and optimize the separability. We usually find appropriate corners with $\sigma = 6$, so we would perhaps need one σ for detecting the corners and one for

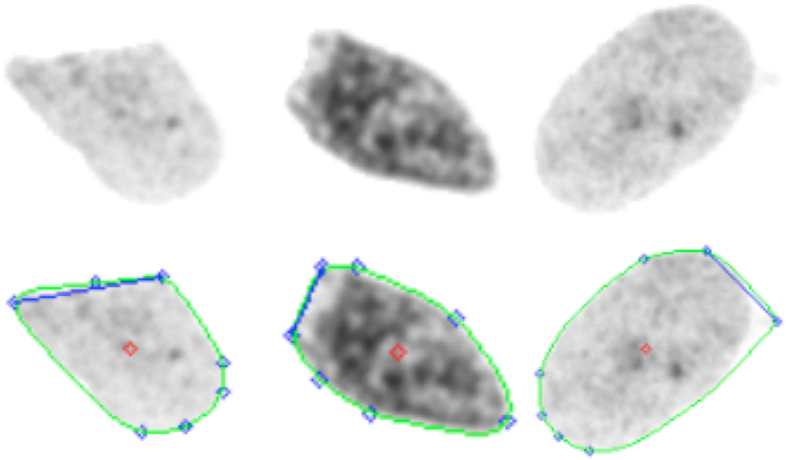


Figure 8.15: To the left is a cell the we judge as clearly cut, the cell in the center is an example of a more questionable cell and the rightmost is a typical example of an over-segmented cell. On the bottom row we have overlaid the region registered as possibly cut, with a blue line. The convex contour is drawn in green, and the points of highest curvature on the cell are drawn as blue circles.

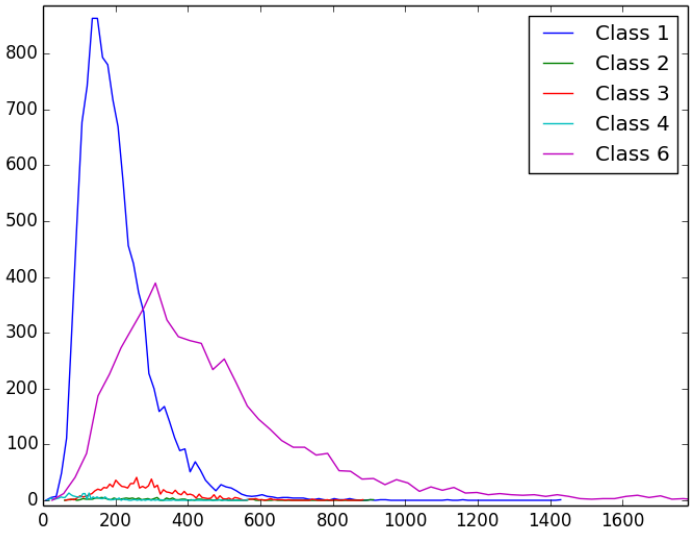


Figure 8.16: Plot of the thresholded PLM13 test-set. The cells removed are mostly correct with a threshold of 520.

Table 8.13: The percentage of cells removed from each class when thresholding the cut cell value on 520.

	PLM13-Train	PLM13-Test
Class 1	2.12 %	0.70 %
Class 2	12.15%	3.50 %
Class 3	10.56%	2.38 %
Class 4	6.25 %	0.0 %
Class 6	25.77 %	22.84 %
Non K6	3.01 %	0.84 %

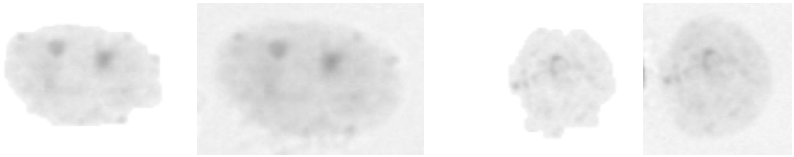


Figure 8.17: Segmented and original images from the L41 set, side by side. The leftmost cell is from K1 and the rightmost from K6. They are often segmented in such a way that they end up with straight edges and sharp corners. There are no clear boundary of what segmentation is acceptable or not.

evaluating the sharpness of a corner.

When developing this features we realized that, no matter how good corners we find, we will still not be able to mimic how humans find cut edges. In order to really improve this feature further, we would need to investigate the contour between the corners, and the remaining contour. Based on the whole contour humans evaluate the cut areas differently. If the roughness differ substantially in an area between two maximum curvature points we are more likely to deem the cell as cut. To utilize this information we tested various approaches with a mapping of the contour to 1D, and then for example try a multiscale Laplace operator, to see if frequencies on the contour differed inside the cut area compared to outside. We also tried to look for differences in entropy of the Fourier transformed curvature. Unfortunately we only succeeded in creating a slow algorithm.

8.2.3 Rough Edges

A large part of the K6 cells have somewhat rough edges. The main problem concerning this features is that some of the large K1 cells with high IOD, also tend to have rough edges. An additional problem with these high IOD cells is that they are often over-segmented, which makes the edges even more rough. With our *mean Fourier spectrum* we also need to watch out for some long and thin K4 cells. Since we do not account for the phase, but only the spectrum, we do not know if all the ellipses are stacked together in the same direction. If all the major axes are rotated in the same or similar direction, we do not get a jagged edge, but rather a very long and thin shape. Examples of both these types of problematic cells are presented in figure 8.19.

We tried to set a threshold above the leftmost cells in figure 8.19, so only

Table 8.14: The percentage of cells removed from each class when thresholding the cut cell value on 520.

	L41-Train	L41-Test
Class 1	1.65 %	2.61 %
Class 2	6.80 %	8.36 %
Class 3	6.98 %	8.62 %
Class 4	1.72 %	2.63 %
Class 6	12.10 %	11.93 %
Non K6	4.68 %	3.09 %

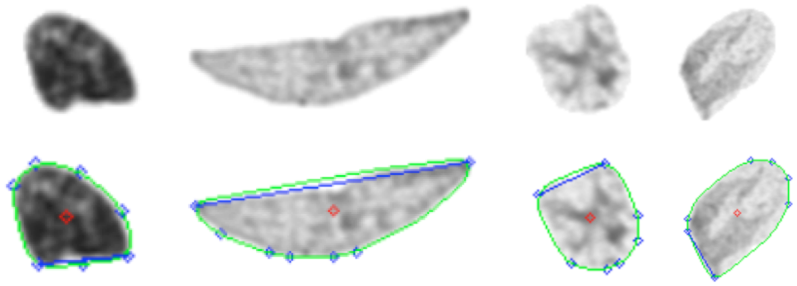


Figure 8.18: Some challenges related to finding cut nuclei. The two leftmost cells are K3 and K4 nuclei. They both possess some attributes that makes them vulnerable to being classified as cut. The K4 cell, second from the left, have extremely high curvature, with an almost straight line between them. The only thing that weights the value down in our algorithm is that the line passes close to the cell center. The two rightmost cells are K6 cells that are hard to detect. One has two slightly rounded corners and the other has one strong and one weak corner. The main problem is that there are many cells with a similar contour that are not cut, so we can not reduce our threshold.

the rightmost cells on both rows were filtered out. In table 8.15 we present the results of this thresholding. Some other cells than K6 were filtered out, especially K4 cells, but these looked either cut, were over-segmented or had rough edges. In figure 8.20, we present two of these cells.

For M51 we have other and and more serious problems. The jaggedness in the M51 cells is often due to under-segmentation, and the main problem is that there seem to be no consensus on what degree of under-segmentation that is acceptable. In this set cells are generally so jagged that the jaggedness due to cuts or a damage cell membrane, become less significant. Indeed our features is not well suited for addressing these problems in the M51 set. We did some research in order to create a feature that measure the relative jaggedness in different parts of the cell contour. It proved to be numerous cells with only partially rough contours also among the K1 cell and the approach was quickly discarded.

We ended up with a high threshold of 0.115, so we could safely separate out the most extreme cases as can be seen in table 8.16. When considering the training-set we noticed that the contour is quite different in these cells.

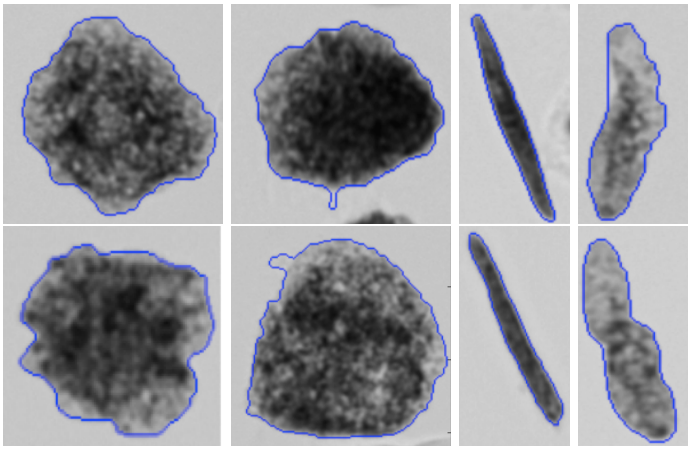


Figure 8.19: Here we display some of the difficulties with setting a threshold. The top row consist of non K6 cells, that have a value for the measure of rough edges, that is just around the threshold value. The bottom row is similar K6 cells, with similar feature values. The two leftmost cells in the top row are high IOD K1 cells, and are especially cells we want to keep in the data set. Still there seems to be some disagreement among the experts also here, as the cells on the row below look very similar in our eyes. The cells in the third column should ideally not be detected by a roughness measure, but as many similar cells is labelled as K6 it may not be a problem.

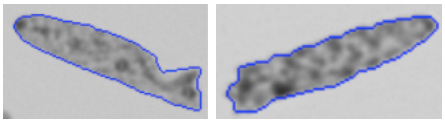


Figure 8.20: Two K4 cells removed from the PLM13 training-set.

These cells had a slightly smoother contour and tended more towards over-segmentation. Still there was many K1 cells with a very jagged edge, and it was generally hard to see the difference between the jagged K1 and K6 cells. With a threshold equal to the one for the PLM13 set we could remove most of these disputed cells, but then we have a quite large loss in non K6 cells. With such a threshold we could remove 12.2% of the K6 cells, but lost 4.19% of the non K6 cells.

For L41 the contours were smoother compared to both M51 sets. The K1 and K4 cells that had ha high value of jaggedness were indeed similar to the jagged cells of K6. We therefore felt that we could safely set the threshold to 0.095, similar to the PLM13 sets. In the test-set slightly more jagged cells were allowed and the consensus on which cells should be excluded seem lower, and this reflects in the thresholding result in table 8.17.

Overview

Differences in segmentation make this feature more suitable for some sets than other. We have at least found a feature that measures the general jaggedness of

Table 8.15: The percentage of cells removed from each class when thresholding the mean Fourier feature value at 0.095.

	PLM13-Train	PLM13-Test
Class 1	0.75 %	0.60 %
Class 2	0.43 %	0.0 %
Class 3	2.93 %	0.08 %
Class 4	3.52 %	7.17 %
Class 6	12.40 %	15.94 %
Non K6	0.93 %	0.68 %

Table 8.16: The M51 datasets, thresholded at a value of 0.115 on the mean Fourier feature.

	M51-Train	M51-Test
Class 1	0.7 %	0.16 %
Class 2	0.0 %	0.0 %
Class 3	0.0 %	0.0 %
Class 4	0.0 %	0.0 %
Class 6	2.94 %	1.27 %
Non K6	0.48 %	0.14 %

a cell contour quite well, but there are also some problems. Especially long and thin cells have a high mean Fourier. The long and thin cells deviates from an elliptic shape, because the thickness are similar along the whole length of the cell. Such a contour needs to be represented by many Fourier coefficients. This problem was unexpected as we did not have any of those cell among the samples we used to develop the feature. A solution could be to use the phase of the Fourier coefficients. We would first have to look at the major and minor axes for each of the ellipses, represented by the Fourier coefficients, $a = \sqrt{|\hat{X}_k| + |\hat{X}_{-k}|}$ and $b = \sqrt{|\hat{X}_k| - |\hat{X}_{-k}|}$. For those pairs of coefficients where b where larger, we could add 90° to the phase. Then the phase would indicate the rotation of the *major axis* of each ellipse. We could then measure whether most of these angles corresponded, as this would indicate that the cell was in fact not jagged, but rather stretched.

Another problem is that in many cases it seems like the relation between smooth and jagged areas are more important than the general jaggedness. As mentioned we tried to create features to tap into that information. We then discovered that the problem is more complex. A K1 cell can often have some areas of extreme curvature, but otherwise have a smooth contour, typically due to over-segmentation. The cells that are excluded typically have a one smooth area and one area with moderate roughness. With more complex phenomenas as this, the feature generation quickly becomes difficult, as the number of cells we can investigate manually are restricted, and we can easily overfit our features.

Table 8.17: The percentage of cells removed from each class when thresholding the mean Fourier feature value at 0.095.

	L41-Train	L41-Test
Class 1	3.52 %	6.42 %
Class 2	0.0 %	0.0 %
Class 3	0.23 %	0.0 %
Class 4	1.73 %	2.64 %
Class 6	16.65 %	15.28 %
Non K6	1.65 %	5.89 %

8.2.4 Over-Segmentation

Over-segmentation is very common, but it has to be allowed within certain limits. Sometimes it is impossible to tell where the segmentation boundary should go, for example due to a grainy texture. For the PLM13 set a threshold of 940 was found to capture many of the clearly over-segmented cells and not remove the cells with only slight over-segmentation. The result of this thresholding can be found in table 8.18. One of the main challenges here is to not remove pale K1 nuclei, or K2 cells in general as they are often over-segmented, but still included. Many cells in PLM13 have a bulge out somewhere along the contour, like the two cell in the center of figure 8.21, some of them are removed and some are not.

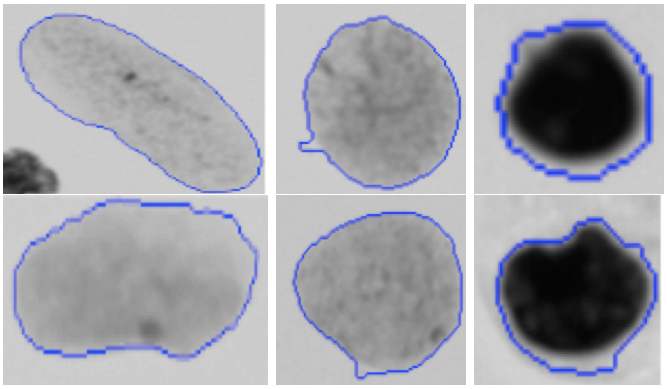


Figure 8.21: Here is a set to illustrate the difficulties in finding a good threshold. The top row are non K6 cells, that are not counted as over-segmented. On the bottom row we have three K6 cells, oversegmented or faded.

The cells in 8.21 we generally want to keep in the set and rather focus on the more clear cases. The PLM13 test-set is similar to the training-set in most regards, but there were slightly more over-segmented K2 and K4 nuclei. We also ended up removing some pale cells with faded edges from K1, similar to the leftmost cells in 8.21. As some of those cells are also in K6 we were not sure how to judge this, but to completely avoid loosing any of the pale cells, we would need to set the threshold to 960, and we would only remove 6.89% of the K6 cells, and not close to the real number of over-segmented cells.

Table 8.18: The percentage of cells removed from each class when thresholding the blurred edge feature at 940

	PLM13-Train	PLM13-Test
Class 1	1.65 %	1.58 %
Class 2	2.51 %	5.52 %
Class 3	0.39 %	1.15%
Class 4	2.78 %	0.93 %
Class 6	14.20 %	14.04 %
Non K6	1.59 %	1.57 %

For the M51 training-set, over-segmentation is not a problem except for K2 cells, which tend to have a surrounding white circle. In this set the tendency is rather towards under-segmentation. Using this feature would therefore do more harm than good, namely removing the K2 before filtering out any K6 cells. We have therefore not included a table for these data-sets. The M51 test-set does in fact have some over-segmented cells, but the over-segmented cells are in all classes and so this feature to detect any K6 cells. In L41 there are also few over-segmented cells, our measure would still have some effect, but this is because in this set they label most of the pale cells as K6. The pale K4 cells are still kept, so to threshold this feature we would lose many of those cells, therefore we do not use this feature on this dataset either.

Overview

Our over-segmentation measure work well for finding the over-segmented regions, but is not a very good measure for how much they are over-segmented. It also dos not use any information from the central part of the cell, so it may be that the cell is not overlapping at all, just very pale. We tested a seeded segmentation with the random walker algorithm [33], where the seeds are place at the location found by our current feature and some pixels in the center of the cell, but it proved hard to find parameters for the algorithm that worked in a general setting. We also tried to scale the feature value depending on the mean intensity, the intensity of the center region or the median intensity. With this scaling we no longer detected the pale cells, but then slightly over-segmented K2 and K3 often became prominent. The main point is that over-segmented cells are often pale, so in scaling by intensity we can no longer detect these cells. The question is also whether re-segmenting the cells might be outside of the main topic of the current thesis.

8.2.5 Combining the Features

We now need to investigate to what degree these features interact. We know that overlapping cells often also are over-segmented. They can also have rough edges and be cut. If all features find almost the same cells our work has been in vain.

We can see that even though they interact to some degree, each feature do provide new information. The results for the PLM13 dataset can be found in table 8.19. For a loss of 3.89% of the non K6, we can remove 42.45% of

Table 8.19: The combined thresholding on all four features

	PLM13-Train	PLM13-Test
Class 1	3.76 %	3.57 %
Class 2	14.57 %	9.82 %
Class 3	11.39 %	5.51 %
Class 4	9.26 %	11.21 %
Class 6	42.45 %	49.13 %
Non K6	4.38 %	3.9 %

the K6 cells. We also have to remember that many of the removed cells could easily ended in K6 or should actually be there. Still more than half the K6 cells are left undetected, so there is still room for improvement. We choose two patients from the PLM13 set and looked through the misclassification. The vast majority of remaining K6 cells, were those similar to K2 and K3. The problem with these cells is that we often cannot see the difference ourself. There may be something related to texture or shape, that we have not understood. Two typically undetected K6 cells are presented in 8.22.

Figure 8.22: The two leftmost cells are typical examples of the large majority of undetected K6 cells. The two rightmost cells are K3 cells, that in our eyes look very similar to the K6 cells.

There were also quite a few cells with some degree of jagged edges or slight over-segmentation that remained undetected, but they are also well represented in the other classes. What is perhaps more severe is that we found some undetected overlapping cells, that unquestionably should have been removed. Most of the "wrongly" removed cells, could be justified to some degree, but here as well the main culprit was the *overlapping detection*. Some K1 cells were clearly not overlapping, but were still removed, as the two cells in the bottom center of figure 8.23.

Table 8.20: Thresholding M51 on the three relevant features: overlapping, cut cell and mean Fourier.

	M51-Train	M51-Test
Class 1	3.31 %	3.52 %
Class 2	6.69 %	15.62 %
Class 3	4.72 %	7.03 %
Class 4	5.9 %	12.75 %
Class 6	20.89 %	27.32 %
Non K6	3.93 %	4.16 %

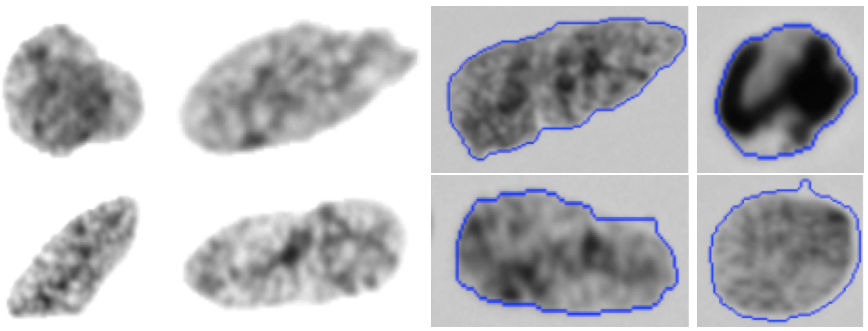


Figure 8.23: The top row are undetected K6 cells, while the bottom row are K1 cells that were detected as K6 cells. The leftmost cell at the top row are quite clearly overlapping, it had a overlap *value of 311.2, which is quite close to the threshold of 320. The two cells in the center of the bottom row were detected as overlapping, where a path was found along the dark grainy pattern.*

Table 8.21: Thresholding L41 on the three relevant features: overlapping, cut cell and mean Fourier.

	L41-Train	L41-Test
Class 1	4.5 %	8.58 %
Class 2	5.91 %	5.98 %
Class 3	3.28 %	3.73 %
Class 4	5.76 %	5.26 %
Class 6	24.8 %	23.76 %
Non K6	4.31 %	8.24 %

For L41 and M51 the undetected K6 cells are more spread across different categories, but many are related to obscure shapes. The results for M51 can be found in the table 8.20 and for L41 they can be found in table 8.21. We could not threshold these sets with the same force as the newer sets, as all the cells could have more jagged edges and obscure shapes in general. The cells falsely detected as K6 were mostly cells with very rough edges, that could be interpreted as both cut and overlapping as well.

8.2.6 Overview of the Thresholding

We created a set of features to fit our visual impression of the different types of cell image debris. Some of the debris could be detected by thresholding these features, but from the results it is evident that there is still much work left before this thresholding techniques can be applied completely automatic, without any human supervision. Not only is there a large part of the K6 cells that we do not fully understand, but even the cells that clearly belong to one of the groups that we have designed features for, can be difficult to detect. With large differences in how well the features work for different datasets, we realize that the features may need to be adapted through the tuning of parameters and adjustments of the thresholds. That process can prove very cumbersome and even unreliable. It may be that the rough filtering that these features provide

can give a sufficiently clean set for most of the later applications, but that is yet to be proved.

As we have learned that we were unable to capture the information needed to separate the K6 cells, we want to see if utilizing these features could help a supervised leaning algorithm to come close to the human level of accuracy for K6. We therefore include a section where we test how well we could classify all the classes, K1-K4 and K6. This means that we loose the level of control we have with the thresholding procedure, but when we do not have the expertise necessary to understand the classification fully, it may be better to seek the aid of the computer.

8.3 Classification of All Classes

Instead of solving this classification as a two-class problem and then later classify the remaining cell-types, we choose to do a supervised classification of all classes. We see no reason for the classification to improve by dividing this process into more stages, as the classifiers create one set of trees for each class, so each class will have different features associated to it. In this section we do the classification without balancing the test-sets in terms class distributions. This is so we can better compare our approach to that of the human experts.

8.3.1 Feature Evaluation

In table 8.22, we present the feature importances for the PLM13 training-set, for classification of all 5 classes. For this classification we see that the feature importance for classification of all classes are slightly more distributed among the features. This is likely because K6 is hard to separate and therefore the classifier uses every possible feature to "slice off" different extreme valued K6. As expected we can se that *cut cell* and *mean Fourier* features are important for the classification. Other typical features that indicating uneven or unsymmetric edges are much more important in this classification, in the evaluation for cell-type classification found in table 8.3. These features are less correlated than the top features from the cell-type classification, as we can see in 8.23. This could partly be due to K6 cells creating "noise", so the features are less related through the class labels. In the evaluation for the features for the M51 classification, found in table 8.24, we see that cut cell is much less important, but area and perimeter is much more important. For the jagged cells in the M51 training-set, we have seen that the effect of the cut cell feature is weakened. For the PLM13 dataset on the other hand we have the problem with large size and color difference in some of the patients. This is probably the reason why *mean Intensity* and *area* have been found so much less effective for separating the cells in PLM13. We have no good explanation for why *IOD* is a so much more effective feature for the PLM13 dataset, but PLM13 *IOD* is clearly one of the more important features, and that may be an issue of concern. We believe that the overlapping feature did not have a high importance, for mainly two reasons. The overlapping feature is a very specific feature and therefore good for classifying only a small part of the K6 cells, and not good for separating among the other classes. There are much more jagged and cut cells, so these features will be chosen first, and many of the overlapping cells have cut or jagged edges

and are therefore removed before the overlapping feature is applied. Despite the fact that it is only good for separating a small part of the cells, we still believe it can be important, due to the effect of the cells the feature remove.

Table 8.22: Feature evaluation for PLM13 training-set, for the evaluation of all classes.

Features	CCR	(RF) Feature Importance	(GBT) Feature Importance
Cut Cell	76.22	0.0979	0.0301
Bending Energy	75.02	0.0829	0.0549
IOD	74.64	0.1894	0.0352
Variance	74.55	0.0647	0.0420
mean Fourier	74.08	0.0771	0.0218
Hu Moment ϕ_1	73.48	0.0350	0.0216
Adaptive GLCM (2 vs. 4)	73.07	0.0369	0.0068
Area	73.06	0.0518	0.0936
Convex Hull Deficiency	72.76	0.0787	0.0312
GLCM-Contrast (d=12)	72.46	0.0167	0.0038
Max Diameter	72.14	0.0394	0.0238
Mean Intensity	72.10	0.0160	0.0146
Perimeter	71.60	0.0356	0.1568
Adaptive GLCM (2 vs. 3)	71.49	0.0118	0.0278
Summed Laplace	70.94	0.0088	0.0392
Entropy	68.63	0.0104	0.0018
Skewness	68.55	0.0099	0.0287
Blurred Edge	68.48	0.0223	0.0261
Circularity	67.27	0.0221	0.0681
Concavity Depth	67.02	0.0247	0.0639
Overlap	66.86	0.0148	0.0128
IOD Balance	65.85	0.0046	0.0111
Eccentricity	65.43	0.0167	0.1079
Distance to Ellipse	65.10	0.0031	0.0067
GLCM-ASM (d=4)	64.63	0.0025	0.0019
Tennengrad Variance	64.52	0.0038	0.0208
Adaptive GLCM (1 vs. 4)	64.44	0.0052	0.0016
Symmetry	64.41	0.0117	0.0294
Jaggedness	64.27	0.0054	0.0161

We have seen that IOD has an important effect on the classification of PLM13, and we therefore want to investigate this effect further. From figure 8.24 we can see that a high IOD increases the probability of a cell to be classified as K1. This is expected as most of the high IOD cells are K1. We had a concern that the extreme value of IOD would indicate a K6 cell, as we do know that this would not be a reliable measure. Fortunately it appears that our model is restricted enough, so that situation will not occur. It is of course also a concern that K1 cells with low IOD will be removed. At least we see that it is more probable that cells with low IOD and smooth edges are classified as

Table 8.23: Correlation coefficients for the combinations of the top 10 features in table 8.22: Cut Cell (CC), Bending Energy (Be), IOD, Variance (V), Mean Fourier (MF), Hu Moment ϕ_1, adaptive GLCM (2 vs. 4) (AG), Area (A), GLCM-Contrast (d=12) (GC).

	CC	BE	IOD	V	MF	HM	AG	A	CHD	GC
CC	1.00	0.55	-0.23	0.35	0.53	-0.20	0.32	-0.41	0.42	0.31
BE	0.55	1.00	-0.31	0.49	0.50	-0.04	0.48	-0.60	0.57	0.50
IOD	-0.23	-0.31	1.00	-0.01	-0.18	-0.11	-0.02	0.42	-0.11	-0.13
V	0.35	0.49	-0.01	1.00	0.32	-0.70	0.83	-0.79	0.25	0.88
MF	0.53	0.50	-0.18	0.32	1.00	-0.03	0.22	-0.33	0.79	0.26
HM	-0.20	-0.04	-0.11	-0.70	-0.03	1.00	-0.61	0.64	0.15	-0.56
AG	0.32	0.48	-0.02	0.83	0.22	-0.61	1.00	-0.71	0.16	0.82
A	-0.41	-0.60	0.42	-0.79	-0.33	0.64	-0.71	1.00	-0.24	-0.72
CHD	0.42	0.57	-0.11	0.25	0.79	0.15	0.16	-0.24	1.00	0.21
GC	0.31	0.50	-0.13	0.88	0.26	-0.56	0.82	-0.72	0.21	1.00

K1, in contrast to jagged cells.

From figure 8.25 we can see that the IOD distribution is quite similar for cells misclassified as K6 and cells correctly classified. Still it seems as though IOD cells have slightly higher probability of begin misclassified. As we saw from the partial dependence plot in figure 8.24, this can not be due to the IOD features itself. It is likely due to the fact that high IOD cells tend to be abnormal in many ways. Still it seems like the bias inflicted on the IOD by the automatic classification is very small. It is hard to determine this without actually applying the further analysis, but with this result we are reassured that the automatic classification is unlikely to affect the ploidy analysis in a major way.

8.3.2 Classification Results

The classification of the PLM13 and M51 sets gave promising results with CCRs of 87.15% and 86.07%, which is similar to the results of our inter-observer study. These results can be found in table 8.25 and 8.26. Still we can see that the precision and recall for the smaller classes are considerably lower than for the larger ones, but this effect is prominent also when comparing the results of the experts. The result for the classification of the inter-observer set is on the other hand more concerning. For the "official labels" of the set we found a CCR of 79.93 %, and the average CCR tested against all the experts was 79.43%. This result is 0.93 % lower than the average accuracy for the worst preforming expert, but within the 0.95 confidence interval ($85.14 \pm 7.29\%$). The full confusion matrix can be found in table8.27. What conclusions to draw from this is uncertain. We know that the data-material in the inter-observer set is significantly different compared to the PLM13 training-set, as we learned in section 8.1. When we know that the relationship between K2 and K3 change dramatically in the set, it may not come as a surprise that these two classes are mixed up also in this classification. The difference in size and variance certainly seems to play a role also here. The worst result is clearly for K4, but this was also very prominent in the inter-observer study, and we have to remember that together any combination of 6 observers did not agree on any of the K4 cells,

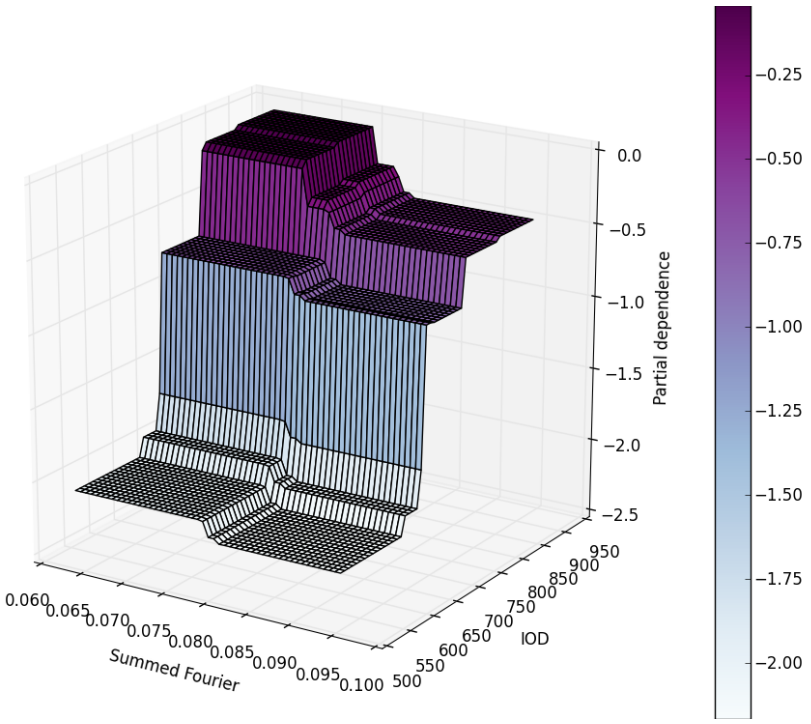


Figure 8.24: Here we can see the partial dependence for the K1 class, between mean Fourier *and* IOD. *We can see that IOD is used for classifying K1 and that it has a much more pronounced effect than the mean Fourier coefficients.*

so these labels are highly unreliable.

With this result we find that we are still undetermined in the question of whether an automatic classification can perform on a par with the manual classification. We do not know whether the PLM13 test-set is generally easier to label, and that the experts would have an even better performance on that set. On the other hand we cannot be sure if the classification algorithm is that much worse either. It may be that the experts are in general more coherent because they had more similar training compared to the expert who labelled the training-set. We still believe that the most probable answer is that the performance of the experts is similar on both sets, but that their method of classification is more robust towards changes in segmentation.

The results for the L41 test-set were markedly worse than for the other sets, as we can see from table 8.28. As we saw earlier there were some differences on some central features in the K2 class, between the training- and test-set. Therefore we wanted to investigate whether we could find similar differences for K6, and calculated the Mahalanobis distances between the features for each class. We found the complete opposite of what we expected as they were in fact quite similar. The largest difference for K6 were for *circularity* with a distance of 0.55 and for K1 the largest difference were in *bending energy* with a value of 0.45. This is not nearly enough to explain the dramatically worse accuracy compared to M51 and PLM13. This led us into an investigation on whether we have overfit our model.

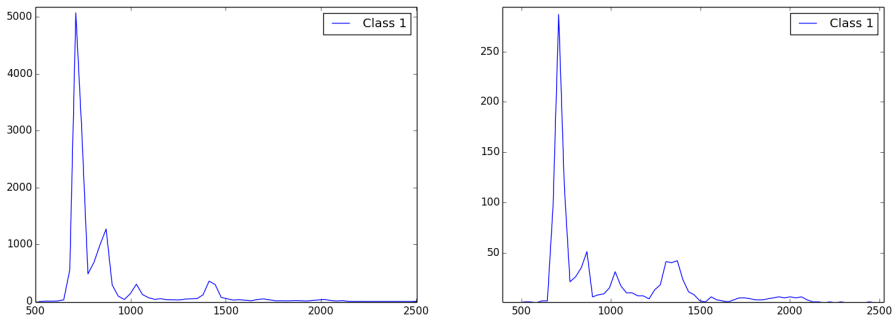


Figure 8.25: The left plot is a histogram of the K1 cells that were correctly classified, while the right plot is the histogram of the wrongly classified K1 cells, from the PLM13-test set. On the x-axis we have the IOD of the cells.

8.3.3 Is The Model Overfitted?

With the poor results of the L41 classification, we have to investigate whether we have created a model so complex that it overfits to the training data, and therefore is less robust to small changes. The results of the other tests indicate no overfitting, but it may be that those pairs of training and test-sets are similar in ways we cannot generally expect. If the model is overfitted we can expect that we would get a very high accuracy when we both train and test on the same set. In doing this we can get an upper bound for how good separation a model could get for a data-set given a "perfect" training-set. Therefore we preformed a classification where we both trained and tested on the L41 test-set, and found a CCR of 80.66%. Which means that we cannot possibly achieve a higher CCR than that given our current model. Since 80.66% accuracy is generally quite low, we could argue that the model may not be complex enough. At least it is clear that we cannot further restrain the complexity, as we then would get an even lower optimal result.

There can be several possible reasons why our model could not achieve any higher CCR on the L41 test-set. It may be that we have not created a set of features that capture the enough of the information that the experts use in their labeling process. We may have restricted the complexity of the classification algorithm too much, so that we are not able to capture some complex interactions of features that the experts use in their labeling. Finally it may be that the underlying error in the labeling is even higher for this data, compared to the other sets.

We investigated how the other data-sets performed when we trained and tested on the same material, and found that M51 had a CCR of 90.39%, PLM13 had 89.03% and the inter-oberver set had 92.34%. Here we have to remember that the ability to separate a data-set with a given model complexity, decreases with the number of cells in the data-set, so therefore the separability of the inter-observer set can be exaggerated, but the test-sets of M51 and PLM13 are both larger than the test-set of L41.

We did a grid search for the L41 training-set to investigate whether underfitting could be a problem and whether increasing the complexity of the classifier

could be the whole solution. In fact the grid search showed that for cross-validation on the L41 training-set, the best tree-depth was 8, instead of 6. With this parameter change, the CCR of the cross-validation changed from 79.8% to 81.35%. With a tree-depth of 8 for the L41 test-set on the other hand the CCR was reduced to 74.86%. The increased complexity obviously also increased the CCR when we trained and tested on the same sets. For the L41 test-set, this score was now 84.33% which is a marked improvement. With the same test and parameters on the other sets we found CCR of 94.16%, 92.05% and 96.05 %. They obviously increased the training accuracy with increased complexity, but their results decreased. Since L41 still has a much worse training accuracy, this leads us to the conclusion that the L41 data-sets are indeed more confounded on our set of features, and no indications of overfitting were found.

8.3.4 Explaining the L41-Result

When we look at the training errors of the two L41 sets, we find that their types of errors are very similar. They mainly confuse K1 and K6 cells, and when investigating the cells further we find that there is especially one type of error that is common. That is that they tend to misclassify jagged cells, one way or the other. In figure 8.26 we can see some examples of the cells that are typically confused. We find that none of the features we have calculated can separate these cells well, and this seems to be the cause of the surprisingly high test error. Since this set was not used much in the generation and search for features it may be that we could have found other features where the differences between these cells were apparent.

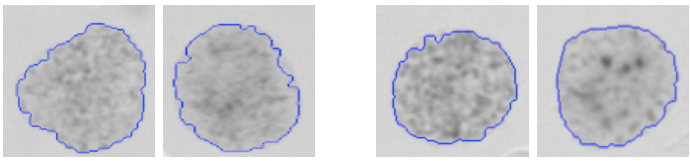


Figure 8.26: Typical problematic cells in L41. The two leftmost cells are K1 cells and the rightmost are K6 cells. They all have rough edges and it is hard to find a systematic difference between them.

This can explain why the L41 tests have a lower CCR than the other sets up to a certain point, but we find that if we do cross-validation on each of the sets we see that there are still a difference between the sets. The cross-validation accuracies of 79.8% for the training-set and 78.8% for the test-set are still markedly better than the test result. The difference between the cross-validation errors and the test errors is that for the test case, we have some additional confusion around the K2 and K3 classes. This has obviously a connection to what we found in the section on the isolated cell-type classification. We find that in the test-set they tend to classify very small black cells as K6 instead of K2. This is a quite common incoherence among all the sets. A more dramatic difference is that they in the test-set have a greater tendency to classify cells that lie between K3 and K1 to K1, while they in the training-set were more prone to label these cells as K3. This can be seen on the Mahalanobis distance of the *area* for the K3 class, between the two sets. The Mahalanobis distance in area is 1.1, and

the K3 cells in the training-set have on average 263 pixels larger area. So larger cells, that were classified as K3 in the training-set are now classified as K1. This have less effect on the mahalanobis distance on the much larger K1 set, but the K1 cells in the test-set are on average 362 pixels smaller.

Table 8.24: Feature evaluation for the M51 training-set,for the evaluation of all classes

Features	CCR	(RF) Feature Importance	(GBT) Feature Importance
Bending Energy	81.82	0.2395	0.1710
Area	80.54	0.0970	0.1142
Perimeter	80.41	0.1031	0.1488
Adaptive GLCM (1 vs. 4)	79.98	0.0719	0.0072
Max Diameter	79.22	0.0532	0.0049
Mean Intensity	78.37	0.0389	0.0004
Variance	77.84	0.0458	0.0088
Adaptive GLCM (2 vs. 4)	76.04	0.0250	0.0013
GLCM-Contrast (d=12)	74.80	0.0342	0.0728
Skewness	73.05	0.0124	0.0144
Adaptive GLCM (2 vs. 3)	71.94	0.0148	0.0013
Summed Laplace	71.50	0.0355	0.0192
Circularity	69.58	0.0132	0.1069
Cut Cell	68.45	0.0152	0.0101
Entropy	67.30	0.0081	0.0076
Concavity Depth	66.38	0.0095	0.0027
GLCM-ASM (d=4)	64.92	0.0088	0.0090
Eccentricity	64.45	0.0103	0.1400
Tennengrad Variance	63.95	0.0120	0.0250
Convex Hull Deficiency	63.05	0.0175	0.0035
Overlap	62.80	0.0082	0.0046
IOD Balance	62.44	0.0065	0.0141
Symmetry	62.31	0.0031	0.0001
Summed Fourier	62.26	0.0208	0.0140
Hu Moment ϕ_1	61.96	0.0526	0.0022
Jaggedness	61.72	0.0127	0.0432
Blurred Edge	61.42	0.0020	0.0017
Distance to Ellipse	60.56	0.0043	0.0138
IOD	58.61	0.0237	0.0372

Table 8.25: Test result for the PLM13 dataset.

Random Forest: PLM13 - Test (18 148) *CCR: 87.15%*

Class	K1	K2	K3	K4	K6	Precision	Recall
K1	14152	79	75	47	1471	0.95	0.90
K2	0	79	42	0	42	0.74	0.48
K3	6	10	912	0	287	0.71	0.75
K4	34	0	0	196	91	0.75	0.61
K6	748	18	250	19	5954	0.76	0.85

Table 8.26: Test result for the M51 dataset.

Random Forest: M51 - Test (17 669) *CCR: 86.07%*

Class	K1	K2	K3	K4	K6	Precision	Recall
K1	10175	0	58	281	409	0.97	0.93
K2	0	32	0	0	0	0.22	1.00
K3	173	0	1130	0	660	0.68	0.58
K4	7	0	0	95	0	0.19	0.93
K6	168	112	466	128	3775	0.78	0.81

Table 8.27: Test result for the inter-observer dataset.

Random Forest: P13 - Inter-Observer (2 989) *CCR: 79.93%*

Class	K1	K2	K3	K4	K6	Precision	Recall
K1	1307	0	0	0	70	0.83	0.95
K2	0	221	100	0	11	0.88	0.67
K3	1	22	443	0	2	0.70	0.95
K4	41	0	0	11	30	1.00	0.13
K6	223	8	92	0	407	0.78	0.56

Table 8.28: Test result for the L41 dataset.

Random Forest: L41 - Test (13 038) *CCR: 75.33%*

Class	K1	K2	K3	K4	K6	Precision	Recall
K1	7871	0	352	0	455	0.82	0.91
K2	0	250	1	0	87	0.65	0.65
K3	46	43	383	0	321	0.22	0.26
K4	35	0	0	0	3	0.00	0.00
K6	1694	79	135	0	1653	0.66	0.46

8.4 Summary of Results

The test results for cell-type classification ranged 91.78-96.64% CCR. For the older datasets L41 and M51, the classification model had a tendency to mislabel cells between K1 and K3, and K1 and K4. For the PLM13 dataset the classification model tended to confuse K3 and K4 nuclei. We found that more K1 cells with low IOD were mislabeled compared to high IOD cells, but the effect was worse when IOD was not included as a feature. The random forest classification model achieved slightly better performance than the gradient boosting model.

We found that for the PLM13 test-set we could remove 49.13% of the K6 cells by a mere thresholding of the features designed to detect debris, with a loss of 3.9% of non K6 cells. These lost cells could by our account also have been classified as K6. For M51 we could remove 27.32% of the K6 cells with a loss of 4.16% of the non K6 cells, with a similar thresholding. For the L41 test-set the result was even worse and we could only remove 23.76% of the K6 cells at a loss of 8.24% of the non K6 cells. The debris detection measures seemed to be somewhat "overfitted" to the newer datasets, as they were developed for the newer datasets and also performed best on those sets.

The test results for the classification of all classes ranged from 75.33-87.15%. For the worst performing dataset, L41, our investigation indicated that the classes were less separable with our feature set, compared to the classes in other datasets. This could either be due to the lack of appropriate features or a generally lower reliability for that dataset.

We find that in general, the performance of the automatic classification is comparable to that of the manual work of humans, but that the automatic classification probably is much easily corrupted by systematic changes in the data-set. The specifically designed features can also be overfitted to a given data-set. We saw that the performance of our novel set of features degraded when tested on the older data-sets, that the features were not designed for. The re-adaption of such features can also be much harder than simply re-training a classification model, but we also have to remember the large amount of work demanded to manually label a new training-set.

Chapter 9

Conclusion and Further Work

With this study we aimed to develop an automatic algorithm for classification of cell-types and removal of debris in form of damaged or overlapping nuclei. We searched through a range of features used in earlier work on classification of cell nuclei and developed a set of novel features explicitly targeted to the task at hand. We classified a part of the PLM13 set into 5 categories with an CCR of 87.15%, which we found to be similar to the results of human experts. The reliability of human experts were tested in a small study on the data from one patient, classified by 7 experts, where they achieved an average CCR of 85.14%. On the same data our classification algorithm had a CCR of 79.43% which is similar to the worst of the human experts and not significantly different from the sample of experts with the average CCR of $(85.14 \pm 7.29\%)$. We found that the data-sets differed greatly in terms of essential features and suggest that the quality of the manual classification may vary to the same degree. Therefore we are inconclusive on whether these results may be generalized to other data-sets.

Making the algorithm robust to different changes in the data material proved difficult as the data differed on many different aspects. This can make our specific features fail for unexpected reasons. We found that creating very specific features can be a rewarding but cumbersome process. Such an approach can also very easily lead to overfitting due to restricted capacity of humans for reviewing large data-sets. We found that our approaches for detecting overlapping, cut and over-segmented cells were partly corrupted in the data-sets where large parts of the samples had irregular contours. A method of more general features that span a large set of different aspects of a cell may therefore be preferred in combination with a powerful classifier. Our study indicate that a supervised classification can adapt to the manual labeling quite well, and perhaps good enough for a full autonomous procedure.

Further work should be put into identifying different sources of changes in the data. For example one could run a test, where human experts tried to label a combined set of data from many different datasets. That way one could learn whether humans really are superior in robustness against changes in the material. If they are, there have to be theoretically possible to develop a set of features that are more robust. If humans are not more robust, we could for example, develop a set of features that include information of attributes related to the slide the images stem from or the patient the tissue is extracted from. The work done on detecting different forms of debris, can advantageously

be incorporated into the segmentation algorithm. As part of the segmentation itself, it may be easier to adapt the features to changes in the segmentation. Detected overlapping cells could perhaps be separated and used, and over- or under-segmented cells could be resegmented.

For cell-type classification, developing a gold-standard in the form of objective staining methods would prove invaluable, but this have proved to be a difficult task. An easier alternative could be to develop a more reliable training-set, by focusing on quality before quantity. This could be done continuously viewing and reviewing a that can be relatively small, but preferably more balance in terms of distribution among the classes.

References

[1] Fritz Albregtsen, Yogesan Kanagasingam, George Farrants, and Havard E Danielsen. Texture discrimination of normal and malignant mouse liver cell nuclei. In *Theory and Applications of Image Analysis: Selected Papers from The 7th Scandinavian Conference on Image Analysis*, pages 324–335, 1992. 80

[2] Fritz Albregtsen, Helene Schulerud, and Luren Yang. Texture classification of mouse liver cell nuclei using invariant moments of consistent regions. In *Computer Analysis of Images and Patterns*, pages 496–502. Springer, 1995. 80

[3] S Baheerathan, Fritz Albregtsen, and Håvard E Danielsen. New texture features based on the complexity curve. *Pattern Recognition*, 32(4):605–618, 1999. 80

[4] Rohit Bhargava, Daniel C Fernandez, Stephen M Hewitt, and Ira W Levin. High throughput assessment of cells and tissues: Bayesian classification of spectral metrics from infrared vibrational spectroscopic imaging data. *Biochimica et Biophysica Acta (BBA) - Biomembranes*, 1758(7):830–845, 2006. 27, 32

[5] Rohit Bhargava, Daniel C Fernandez, Stephen M Hewitt, and Ira W Levin. High throughput assessment of cells and tissues: Bayesian classification of spectral metrics from infrared vibrational spectroscopic imaging data. *Biochimica et Biophysica Acta (BBA)-Biomembranes*, 1758(7):830–845, 2006. 29, 30

[6] Christophe Boudry, Paulette Herlin, Benoit Plancoulaine, Eric Masson, Abderrahim Elmoataz, Hubert Cardot, Michel Coster, Daniel Bloyet, and Jean-Louis Chermant. Automatic morphological sieving: comparison between different methods, application to DNA ploidy measurements. *Analytical Cellular Pathology*, 18(4):203–210, 1999. 29, 31

[7] Christophe Boudry and Brigitte Sola. Mathematical Morphology: Application to DNA Ploidy. *Microsc. Microanal. Microstruct*, 7(1996):477, 1996. 29

[8] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. 93, 107, 108, 109, 112

[9] Robert L Brennan and Dale J Prediger. Coefficient kappa: Some uses, misuses, and alternatives. *Educational and psychological measurement*, 41(3):687–699, 1981. 15

[10] Patrick Brigger, Jeff Hoeg, and Michael Unser. B-spline snakes: a flexible tool for parametric contour detection. *Image Processing, IEEE Transactions on*, 9(9):1484–1496, 2000. 27, 131

[11] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM, 2006. 109, 127

[12] Antonin Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical imaging and vision*, 20(1-2):89–97, 2004. 53

[13] Richard W Conners, Mohan M Trivedi, and Charles A Harlow. Segmentation of a high-resolution urban scene using texture operators. *Computer Vision, Graphics, and Image Processing*, 25(3):273–310, 1984. 83

[14] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, 1967. 94

[15] Glenn De'Ath. Boosted trees for ecological modeling and prediction. *Ecology*, 88(1):243–251, 2007. 101

[16] Andrew G Dempster and Cecilia Di Ruberto. Using granulometries in processing images of malarial blood. In *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*, volume 5, pages 291–294. IEEE, 2001. 90

[17] Cecilia Di Ruberto, Andrew Dempster, Shahid Khan, and Bill Jarra. Analysis of infected blood cell images using morphological operators. *Image and Vision Computing*, 20(2):133–146, February 2002. 90

[18] Thomas G Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, 40(2):139–157, 2000. 101, 109

[19] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959. 53

[20] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973. 45

[21] Dominik Maria Endres and Johannes E Schindelin. A new metric for probability distributions. *Information Theory, IEEE Transactions on*, 49(7):1858–1860, 2003. 72

[22] A. Fitzgibbon, M. Pilu, and R.B. Fisher. Direct least square fitting of ellipses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21, 1999. 33, 34

[23] Joseph L Fleiss and Jacob Cohen. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and psychological measurement*, 1973. 15

[24] Jan Flusser. On the independence of rotation moment invariants. *Pattern recognition*, 33(9):1405–1410, 2000. 88

[25] Herbert Freeman. Boundary encoding and processing. *Picture processing and psychopictorics*, 241, 1970. 65

[26] Yoav Freund and Robert E Schapire. Experiments with a new boosting algorithm. In *ICML*, volume 96, pages 148–156, 1996. 103

[27] Brendan J Frey and Delbert Dueck. Mixture modeling by affinity propagation. *Advances in neural information processing systems*, 18:379, 2006. 75

[28] Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007. 75

[29] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000. 93, 103, 108

[30] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001. 93, 105, 106, 107, 109

[31] David L Fritzsche. A systematic method for character recognition. Technical report, 1961. 34

[32] Rafael C Gonzalez and Richard E Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006. 35

[33] Leo Grady. Random walks for image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(11):1768–1783, 2006. 141

[34] Gösta H Granlund. Fourier preprocessing for hand print character recognition. *Computers, IEEE Transactions on*, 100(2):195–201, 1972. 34, 35

[35] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003. 98

[36] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, 1982. 5

[37] Robert M Haralick, Karthikeyan Shanmugam, and Its' Hak Dinstein. Textural features for image classification. *Systems, Man and Cybernetics, IEEE Transactions on*, (6):610–621, 1973. 31, 83

[38] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988. 47

[39] Trevor Hastie, Robert Tibshirani, Jerome Friedman, T Hastie, J Friedman, and R Tibshirani. *The elements of statistical learning*, volume 2. Springer, 2009. 93, 100, 101, 104, 106, 107, 108, 109

[40] David W Hedley. DNA analysis from paraffin-embedded blocks. *Methods in cell biology*, 41:231–240, 1994. 8

[41] Robert C Holte. Very simple classification rules perform well on most commonly used datasets. *Machine learning*, 11(1):63–90, 1993. 108

[42] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, 1962. 88

[43] John D Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. 61, 91

[44] Daniel P Huttenlocher, Gregory A Klanderman, and William J Rucklidge. Comparing images using the Hausdorff distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(9):850–863, 1993. 45

[45] Itzees. Open Source Computer Vision Library. 34, 52, 61, 91

[46] Karen E Jacowitz and Daniel Kahneman. Measures of anchoring in estimation tasks. *Personality and Social Psychology Bulletin*, 21:1161–1166, 1995. 20

[47] Gaetano Kanizsa. Subjective contours. *Scientific American*, 234(4):48–52, 1976. 52

[48] G B Kristensen, W Kildal, V M Abeler, J Kaern, Ignace Vergote, C G Trope, and H E Danielsen. Large-scale genomic instability predicts long-term outcome for women with invasive stage I ovarian cancer. *Annals of oncology*, 14(10):1494–1500, 2003. 19, 22

[49] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, pages 79–86, 1951. 72

[50] J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977. 16

[51] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 4

[52] Jianhua Lin. Divergence measures based on the Shannon entropy. *Information Theory, IEEE Transactions on*, 37(1):145–151, 1991. 72

[53] Richard Maclin and David Opitz. Popular ensemble methods: An empirical study. *arXiv preprint arXiv:1106.0257*, 2011. 103

[54] J Maddison. *Digital image processing for prognostic and diagnostic clinical pathology*. PhD thesis, University of Huddersfield, 2005. 8, 67, 69

[55] Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and Sašo Džeroski. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, 45(9):3084–3104, September 2012. 107

[56] Aamir Saeed Malik. Comparison of focus measures under the influence of various factors effecting their performance. *Depth Map and*, 2012. 60

[57] A G Mamistvalov. n-dimensional moment invariants and conceptual mathematical theory of recognition n-dimensional solids, 1998. 88

[58] Eric Masson, Paulette Herlin, Isabelle Galle, Françoise Duigou, Philippe Belhomme, Daniel Bloyet, and Anne-Marie Mandard. Automatic classification of cellular elements of solid tumors: application to DNA quantitation. *Acta Stereologica*, 13(1):75–81, 1994. 29, 31

[59] Birgitte Nielsen, Fritz Albregtsen, Sivalingam Baheerathan, and Havard E Danielsen. Peel-off-scanning to obtain radial differentiation of fractal and complexity features in cell nuclei. In *Proceedings, Vision Interface*, pages 54–60, 2000. 72

[60] Birgitte Nielsen, Fritz Albregtsen, and Håvard E Danielsen. The use of fractal features from the periphery of cell nuclei as a classification tool. *Analytical Cellular Pathology*, 19(1):21–37, 1999. 90

[61] Birgitte Nielsen, Fritz Albregtsen, and Havard E Danielsen. Statistical nuclear texture analysis in cancer research: A review of methods and applications. *Critical Reviews in Oncogenesis*, 14(2-3), 2008. 19, 25, 81

[62] Birgitte Nielsen, Fritz Albregtsen, Wanja Kildal, Vera M Abeler, Gunnar B Kristensen, and Håvard E Danielsen. The prognostic value of adaptive nuclear texture features from patient gray level entropy matrices in early stage ovarian cancer. *Analytical Cellular Pathology*, 35(4):305–314, 2012. 8, 19, 23

[63] Birgitte Nielsen, Fritz Albregtsen, Wanja Kildal, and Håvard E Danielsen. Prognostic classification of early ovarian cancer based on very low dimensionality adaptive texture feature vectors from cell nuclei from monolayers and histological sections. *Analytical Cellular Pathology*, 23(2):75–88, 2001. 19, 80

[64] Birgitte Nielsen and Håvard E Danielsen. Prognostic value of adaptive textural features-The effect of standardizing nuclear first-order gray level statistics and mixing information from nuclei having different area. *Analytical Cellular Pathology*, 28(3):85–95, 2006. 19

[65] Birgitte Nielsen and Håvard E Danielsen. Prognostic value of adaptive textural features-The effect of standardizing nuclear first-order gray level statistics aNielsen, B., & Danielsen, H. E. (2006). Prognostic value of adaptive textural features-The effect of standardizing nuclear first-order gr. *Analytical Cellular Pathology*, 28(3):85–95, 2006. 82, 85

[66] R A Olshen, L Breiman, J H Friedman, and Charles J Stone. Classification and regression trees. *Wadsworth International Group*, 1984. 93, 99, 101

[67] José Luis Pech-Pacheco, Gabriel Cristóbal, Jesús Chamorro-Martinez, and Joaquín Fernández-Valdivia. Diatom autofocusing in brightfield microscopy: a comparative study. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 3, pages 314–317. IEEE, 2000. 60

[68] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, and Vincent Dubourg. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011. 61, 109

[69] Said Pertuz, Domenec Puig, and Miguel Angel Garcia. Analysis of focus measure operators for shape-from-focus. *Pattern Recognition*, 46(5):1415–1432, 2013. 60

[70] Said Pertuz, Domenec Puig, and Miguel Angel Garcia. Analysis of focus measure operators for shape-from-focus. *Pattern Recognition*, 46(5):1415–1432, 2013. 60

[71] John Ross Quinlan. *C4. 5: programs for machine learning*, volume 1. Morgan kaufmann, 1993. 99

[72] Prabhu Ramachandran and Gaël Varoquaux. Mayavi: 3D visualization of scientific data. *Computing in Science & Engineering*, 13(2):40–51, 2011. 91

[73] Richard A Redner and Homer F Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM review*, 26(2):195–239, 1984. 94

[74] Natsuki Sano, Hideo Suzuki, and Masato Koda. A robust boosting method for mislabeled data. *Journal of the Operations Research Society of Japan-Keiei Kagaku*, 47(3):182, 2004. 103, 104

[75] Helene Schulerud. Bias of error rates in linear discriminant analysis caused by feature selection and sample size. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 2, pages 372–377. IEEE, 2000. 25

[76] Helene Schulerud and Fritz Albregtsen. Many are called, but few are chosen. Feature selection and error estimation in high dimensional spaces. *Computer methods and programs in biomedicine*, 73(2):91–99, 2004. 98

[77] Jack Sklansky. Finding the convex hull of a simple polygon. *Pattern Recognition Letters*, 1(2):79–83, 1982. 66

[78] Johan A K Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999. 94

[79] Amir Tahmasbi, Fatemeh Saki, and Shahriar B Shokouhi. An effective breast mass diagnosis system using zernike moments. In *Biomedical Engineering (ICBME), 2010 17th Iranian Conference of*, pages 1–4. IEEE, 2010. 89

[80] H J Tanke and E M Van Ingen. A reliable Feulgen-acriflavine-SO2 staining procedure for quantitative DNA measurements. *Journal of Histochemistry & Cytochemistry*, 28(9):1007–1013, 1980. 8

[81] Godfried T Toussaint. Solving geometric problems with the rotating calipers. *Proc. IEEE Melecon*, 83:A10, 1983. 66

[82] Oeivind Due Trier and Anil K Jain. Goal-directed evaluation of binarization methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(12):1191–1201, 1995. 4

[83] Stéfan van der Walt, S Chris Colbert, and Gaël Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science & Engineering*, 13(2), 2011. 61, 91

[84] Stéfan van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: Image processing in Python. Technical report, 2014. 61

[85] Vn Vapnik. the nature of Statistical Learning theory. *New Yorc Springer Verlag*, 1995. 93

[86] R F Walker, P T Jackway, and I D Longstaff. Recent developments in the use of the co-occurrence matrix for texture recognition. In *Digital Signal Processing Proceedings, 1997. DSP 97., 1997 13th International Conference on*, volume 1, pages 63–65. IEEE, 1997. 82

[87] Ross F Walker, Paul Jackway, Brian Lovell, and I D Longstaff. Classification of cervical cell nuclei using morphological segmentation and textural feature extraction. In *Intelligent Information Systems, 1994. Proceedings of the 1994 Second Australian and New Zealand Conference on*, pages 297–301. IEEE, 1994. 80

[88] Baolin Wu, Tom Abbott, David Fishman, Walter McMurray, Gil Mor, Kathryn Stone, David Ward, Kenneth Williams, and Hongyu Zhao. Comparison of statistical methods for classification of ovarian cancer using mass spectrometry data. *Bioinformatics*, 19(13):1636–1643, 2003. 107

[89] Thomas Würflinger, Jens Stockhausen, Dietrich Meyer-Ebrecht, and Alfred Böcking. Robust automatic coregistration, segmentation, and classification of cell nuclei in multimodal cytopathological microscopic images. *Computerized Medical Imaging and Graphics*, 28(1):87–98, 2004. 27, 28, 52

[90] Charles T Zahn and Ralph Z Roskies. Fourier descriptors for plane closed curves. *Computers, IEEE Transactions on*, 100(3):269–281, 1972. 34

[91] Michael Zambon, Rick Lawrence, Andrew Bunn, and Scott Powell. Effect of alternative splitting rules on image processing using classification tree analysis. *Photogrammetric Engineering and Remote Sensing*, 72(1):25, 2006. 100

[92] T Y Zhang and Ching Y Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, 1984. 66