TAMPERE UNIVERSITY OF TECHNOLOGY

MASI VALKONEN

TRAINING BASED SEGMENTATION OF ELECTRON MICROSCOPE IMAGES

Master's thesis

Examiner: Heikki Huttunen
Supervisors: Minnamari Vippola and Erkki Levänen
Examiners and topic approved by the Council of the Faculty of Computing and
Electrical Engineering on 3 June 2015

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Signaalinkäsittelyn ja tietoliikennetekniikan koulutusohjelma
VALKONEN, MASI: Elektronimikroskooppikuvien segmentointi oppivan järjestelmän avulla
Diplomityö, 64 sivua
Toukokuu 2015
Pääaine: Multimedia
Tarkastajat: Heikki Huttunen
Avainsanat: Logistinen regressio, LASSO regularisointi, watershed, segmentointi, huokonen, partikkeli
Nykypäivänä materiaalitukimuksen kuva-analyysi toteutetaan vielä suurelta osin manuaalisesti, mikä on usein työlästä ja aikaa vievää. Tässä työssä tutkitaan kahta materiaalitieteen yleistä kuva-analyysiongelmaa, joita varten kehitetään automaattiset menetelmät helpottamaan analyysiä.

Ensimmäinen tapaus käsittelee zeoliittirakenteen huokosten segmentointia logistisen regression ja LASSO regularisoinnin avulla. Menetelmä toimii johdonmukaisemmin ja nopeammin kuin ihminen ja löytää lisäksi enemmän huokosia. Suorituskyvyksi saatiin PAS metriikalla mitattuna 0.79 ja F_1 metriikalla 0.89. Tuloksissa iso osa virheestä johtuu siitä, että ihminen voi löytää huokosia monella eri tavalla mutta menetelmä vain yhdellä.

Toisessa tapauksessa hopananopartikkeleita segmentoidaan elektronimikroskooppikuvista käyttäen LASSO regularisoitua logistista regressiota ja watershed menetelmää. Menetelmä on nopeampi kuin ihminen ja tuottaa hyviä tuloksia kun partikkelit eivät ole paljon limittäin. Päällekkäiset hiukkaset ovat kuitenkin vaikeita erotella, minkä vuoksi tulokset eivät ole yhtä hyviä. Suorituskyky PAS metriikalla mitattuna on 0.76 ja F_1 metriikalla 0.86.

Oppiva järjestelmä yhdistettynä automaattiseen piirteen valintaan on siitä mielenkiintoinen, että opetusdatan voi kerätä kuka tahansa eikä pohjatietämystä kuvankäsittelystä tarvita. Tämä mahdollistaa yleiskäyttöisen segmentoinitohjelman kehittämisen, joka osaa mukautua ongelmaan kerätyn datan perusteella.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Signal Processing and Communications Engineering
VALKONEN, MASI: Training based segmentation of electron microscope images
Master of Science Thesis, 64 pages
May 2015
Major: Multimedia
Examiner: University Lecturer Heikki Huttunen
Keywords: Logistic regression, LASSO regularization, watershed, segmentation, pore, particle

A great deal of image analysis in today's materials research is done manually, which can be time consuming and tedious. This thesis is a case study of two image analysis problems in the field of materials science for which automatic methods are developed to aid the analysis process.

In the first case, an automatic segmentation method is developed to segment zeolite pores. The method is based on logistic regression combined with sparsity promoting LASSO regularization. It is able to find more pores than humans with better consistency and speed yielding average PAS score of 0.79 and F_1 score of 0.89. Much of the error is caused by the fact that humans specify the pore perimeters in various ways, but the method consistently produces similar segmentations.

The second case considers automatic segmentation of silver nanoparticles by combining LASSO regularized logistic regression and watershed segmentation. The method is faster than manual segmentation and produces fine segments when particles have little overlap. However, if the amount of overlap is high, the segments are flawed. The performance of the system in terms of PAS metric and F_1 score is 0.76 and 0.86, respectively.

An interesting property of training based segmentation together with sparsity promoting property is that training data can be collected by anyone. This enables creating an adaptive segmentation software for anyone regardless of image processing experience.

PREFACE

After thousands lines of Matlab code, 12 months and over 20 000 written words, I am glad to be finally writing this chapter. Many highs and lows have been part of this experience, which wouldn't have been possible without the guidance and help from others. I would like to sincerely thank university lecturer Heikki Huttunen for providing his support during the writing process of this thesis. He has given me as much freedom as I have ever wanted and allowed me to experiment with ideas that were sometimes close to nonsense. I have really enjoyed my time under his supervision and hope that I may have an opportunity to work with him in the future.

I would also like to thank Matti Järveläinen with whom I had multiple long conversations. He always made the time for my questions and shared his view on research work. Also, Minnamari Vippola and Hanna Silen deserve thanks for offering their help.

Annotating was one of the most tiresome parts of this work and therefore I would like to thank Johannes Lehmusvaara, Richard Terkovič, Teemu Vartiainen and Nyyti Kinnunen for creating many of the ground truth images. Some of you were not even obligated to do so which I greatly appreciate.

Perhaps the most supportive has been my dear girlfriend Inari who was able to endure me even during the most difficult times. In any case, I hope the challenges are far from over since often the most difficult things are also the most rewarding.

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

SEM	Scanning electron microscope
TEM	Transmission electron microscope
LASSO	Least Absolute Shrinkage and Selection Operator
HE	Histogram equalization
AHE	Adaptive histogram equalization
CLAHE	Contrast limited adaptive histogram equalization
λ	Regularization parameter for logistic regression
TPR	True positive rate
FPR	False positive rate
CV	Cross-validation
ML	Maximum likelihood

1. INTRODUCTION

Nanomaterials have received a lot of research focus over the last few years due to their unique properties present only at nanoscale. Because the scale is extremely small, analysis of such materials is often supported by microscopic imaging. Specifically electron microscopy allows imaging at higher magnifications since the limitations of traditional light microscopy do not apply when using an electron beam [1].

To better understand commonalities and differences in the materials, researchers need objective measures of phenomena in the images. For example, one might want to know the average particle size or obtain the probability distribution of orientation of pores in the image. A typical way to achieve this would be extracting each object in the image from the background, which makes these problems fall into the category of image segmentation.

Although software and processing power of modern computers have advanced rapidly, much of the measuring is still accomplished by manually indicating where the object boundaries are. Manually processing a set of images can be tedious and time consuming. Moreover, the quality of work depends heavily on the person, which makes it subjective and inconsistent. This process can be aided with automatic tools developed in signal processing that provide faster and more consistent quality of work.

In this work, two example cases are introduced and automatic segmentation procedures are developed using training based methods. The first case considers segmentation of material structures made of zeolite, and the second case studies segmentation of silver nanoparticles. In a training based method, a computer is shown examples of interesting objects in the images from which an algorithm learns their general appearance. With the aid of the algorithm, new images are segmented automatically. This work focuses mainly on image segmentation since calculating parameters of the objects in an image becomes a straightforward task once segmentation has been performed.

The first case studies segmentation of microscope images of zeolite, which has been researched for gas separation processes such as carbon dioxide capturing [2]. One manufacturing method comprises freeze-casting and sintering zeolite 13X powder, bentonite, and polyethylene glycol resulting in brittle structure that is not convenient for industrial use. There is ongoing research on which factors contribute

the most to tensile and compression strengths of the material. It has been found that the tensile strength peaks using certain manufacturing process in which the aspect ratio of pores has been found to hold significance. In this work, an automatic procedure is developed to separate individual pores from the rest of the image.

The second case studies segmentation of silver nanoparticles, which is an active field of research in materials science. Nanosilver has been examined for its ability to kill bacteria more efficiently than microsilver using notably less amount of matter [3]. For this reason, it has been applied as a coating for medical devices to reduce the risk of device based infections. The potency of nanoparticles is explained by higher surface-area-to-volume ratio because surface atoms are less stabilized and thus react more easily [4]. Since approximation of the surface area is a tedious and repeating task, an automatic segmentation procedure is developed to determine the area of each individual silver nanoparticle in a TEM image.

Related topics have been researched before. Ruusuvuori *et al.* [5] used LASSO regularized logistic regression classifier along with Markov random field spatial prior to segment microscope images of subcellular objects. Liu *et al.* [6] applied watershed merge tree method to segment electron microscope images of neural tissue. In the work, a random forest classifier was trained to predict the merging probability of two regions after which the final segmentation result is given by the highest region potential. Wong *et al.* [7] applied a variety of thresholding methods on electron microscope images of cement-based material to segment pores from the material.

The structure of this work is as follows. In Chapter 2 the basic idea of image segmentation is introduced. Chapter 3 explains the theory behind each method that is used in the procedures in Chapters 4 and 5. Conclusions of this work are presented in Chapter 6.

2. IMAGE SEGMENTATION

Image segmentation is an important and one of the most difficult problems in image analysis [8]. Because it is one of the first steps in an automatic image analysis system, and subsequent steps depend on it, it has a major effect on the overall analysis performance. One definition of segmentation is to divide an image into homogeneous regions that are inhomogeneous with any adjacent region. This means, in principle, partitioning the image into meaningful regions where the content has similar attributes. An example image of segmentation is shown in Figure 2.1, where the image has been divided into two distinct regions, foreground and background. Pixels that belong to the foreground are denoted by value 1 (white) and pixels in the background are denoted by value 0 (black).

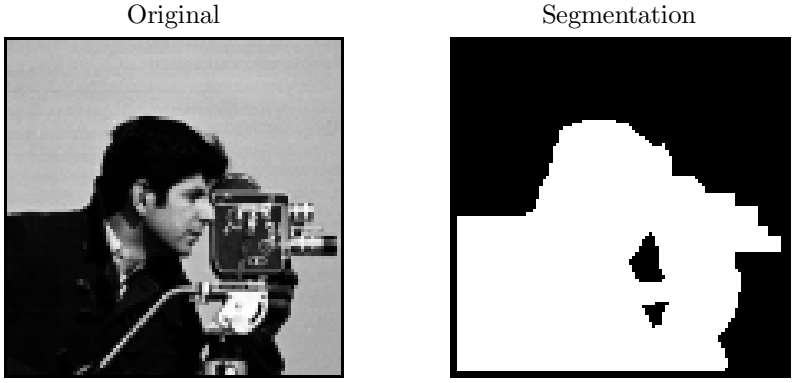

Figure 2.1: A visualization of image segmentation. Here the object of interest, the photographer, has been separated from the background

One interpretation of image segmentation is that these homogeneous regions describe objects of higher meaning in the image. As digital images are represented only using very low level elements, pixels, image segmentation can be viewed as a transition from pixel level to object level. [9]

Segmentation Pipeline

This work focuses on studying and solving two real-world segmentation problems with training based segmentation methods. The methods presented in this work follow the steps shown in Figure 2.2, where blocks drawn in solid line are within the scope of this thesis. The framework comprises two phases, training phase and

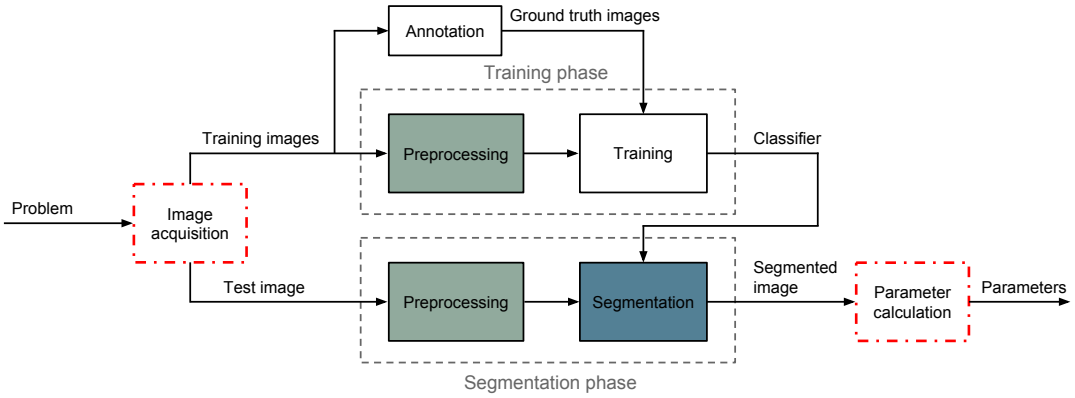

Figure 2.2: A block diagram illustrating the general structure of the developed segmentation procedures. The blocks drawn in solid lines are discussed in this thesis. The blocks drawn in red dash-dot line are part of a functional image analysis system but are not considered in this work.

segmentation phase, denoted by dashed gray line. In training phase, an algorithm, also known as a classifier, is trained to distinguish objects of interest in the images. Training phase is followed by segmentation phase, where new images are segmented with the aid of the trained classifier. Because a trained classifier is needed in segmentation phase, it is required to complete the training phase first. Once training phase is completed, segmentation phase may be executed as many times necessary.

The process of creating a complete image segmentation system begins with a problem that needs to be solved automatically. The problems in this thesis concern extraction of different parameters of objects in material samples, *e.g.*, calculating the average particle size of a silver nanoparticle or measuring the aspect ratio of a pore in a zeolite sample.

The first block in the diagram is image acquisition in which material samples, also known as specimens, are imaged using an electron microscope. This stage outputs images that hold necessary information to solve the problem. Two distinct sets of images are collected during this stage. One set contains images the classifier is trained with, *i.e.*, training images, and the other set holds the test images for which the actual segmentation is performed.

In supervised learning, a classifier learns the appearance of interesting objects in the images. The locations of such objects and their corresponding categories need to be specified in the images. In many cases, this information is not readily available but needs to be added manually to the data. The process of manually adding *labels* to the data is called *annotation*, which is shown as a separate block in the diagram. The resulting ground truth images are then used together with the training image data to train a classifier. Supervised learning is further considered in Chapter 3.4.

Image acquisition step rarely produces images that are usable as such in training

phase or segmentation phase. Therefore a preparation step called preprocessing is needed [10]. Preprocessing is an important step in automatic image analysis system where essential information is distilled. It sometimes means discarding irrelevant information to make segmentation task easier [11]. Preprocessing can modify the data in such way that the outcome is not easily readable by humans, but in better form for the following segmentation method. Examples of preprocessing steps include contrast enhancement and noise removal.

The last block in segmentation phase is the actual segmentation step, where test images are partitioned into meaningful regions with the aid of trained classifier. In practice, the produced regions are objects in the images, from which desired object parameters are computed. For example, the area of an object is calculated simply by counting the number of pixels in a region and converting the value to physical world units such as square nanometers.

3. METHODS

Automatic image analysis systems are typically built by combining multiple different methods rather than applying one single method to solve a problem. There is no guaranteed order in which these methods appear in the final system and some of them may be used more than once in different parts of the processing pipeline [12]. Therefore, to avoid redundancy in the following chapters, the most important methods are explained here in detail.

3.1 Histogram Equalization

Microscope imaging is a challenging task where many parameters and prior treatment of specimens have an effect on the produced image. It is not unusual for the images to have imperfections that have harmful effects on automatic segmentation methods. Examples of such imperfections are lack of contrast and its variation across an image. [13]

Histogram Equalization (HE) is a method to enhance contrast in an image by applying a transform $T(\cdot)$ for each pixel in the image. In theory, the method produces uniformly distributed histogram, but in practice, only close to flat histograms are achieved [14]. Forcing uniformly distributed histogram renders some pixels black and some white, which often increases contrast.

The transform is based on histogram, which is a representation of an image expressing the distribution of gray levels in the image. It is defined as

$$H(r_k) = n_k, \tag{3.1}$$

where r_k is kth gray level and n_k is the number of pixels having gray level r_k in the image. The number of possible gray levels is L and therefore $k \in \{1, \ldots, L\}$. For example, if images are represented using 8-bit integers, possible number of different gray levels is $L = 2^8 = 256$.

The actual transform uses cumulative histogram

$$C(r_k) = \sum_{j=0}^{k} H(r_j), \tag{3.2}$$

which is the sum of histogram values up to a certain gray level. Now each pixel in

the image may be transformed by

$$J(i,j) = T(I(i,j)) = (L-1)\frac{C(I(i,j))}{MN}, \tag{3.3}$$

where M and N are the dimensions of the input image. Dividing the histogram by MN scales the cumulative histogram to have range $[0,1]$, and multiplying by $L-1$ scales the output value to the desired range. In this case, the transform function is monotonically increasing, which means that the order of gray levels is preserved in the transform. In other words, the darkest gray levels in the image are mapped to the darkest values also in the result image. [14]

The amount of contrast enhancement applied to a certain gray level depends on the slope of the transform function. This is because slope determines the ratio between a range of intensities in the input image and a range of intensities in the output image. Slopes above 1 denotes mapping a smaller range of intensities from the input image to a larger range of intensities in the output image, leading to increased contrast for that specific range of intensities. Following the same logic, slopes below 1 compresses the intensity range and decrease contrast in the output image. This method is implemented In MATLAB as the function `histeq` that is part of the image processing toolbox. [14]

A visualization of it is shown in Figure 3.1, where the bottom left image is the

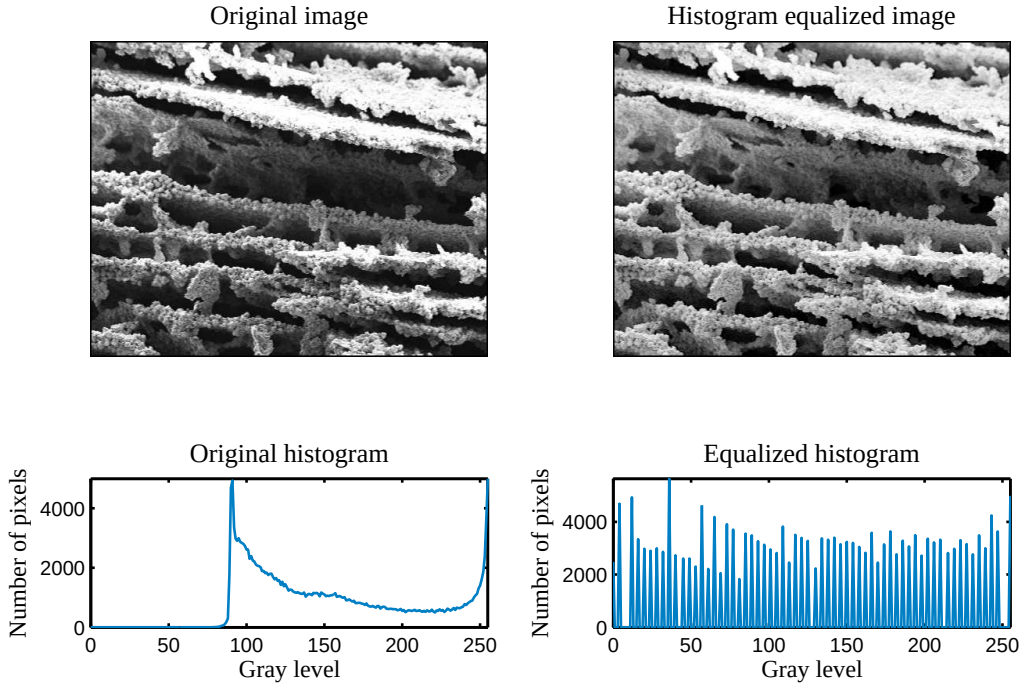

Figure 3.1: An example of histogram equalization. In the top-left corner is the original unprocessed image. In the top-right corner is the histogram equalized image. Below these images are the corresponding histograms.

histogram of the original image (top left). The histogram shows that the numeric range is not entirely used as the darkest value is around 80. The bottom right image shows the equalized histogram, where the envelope of the graph is more even, and the full range of possible intensities is now in use.

3.2 Contrast Limited Adaptive Histogram Equalization

Histogram equalization processes the image globally, *i.e.*, it uses the same transform function throughout the image that sometimes results in locally poor contrast. Adaptive histogram equalization (AHE) tries to overcome this by using different transform functions in different portions of the image. There are various ways to accomplish this, and in this section one such approach is introduced.

The algorithm operates by dividing the input image into equally sized blocks and for each block histogram equalization is performed. The separately processed blocks are then combined back together to produce the final image. However, this method has two major shortcomings. Processing each block as if they were individual images may lead to visible transition lines at block boundaries since each block is processed independently of the neighboring blocks. Additionally, if one block contains only values from a homogeneous region, where variation of intensities is low, contrast enhancement for such region becomes excessive [15]. These problems are depicted in Figure 3.2, where the top row contains patches from the original image and the

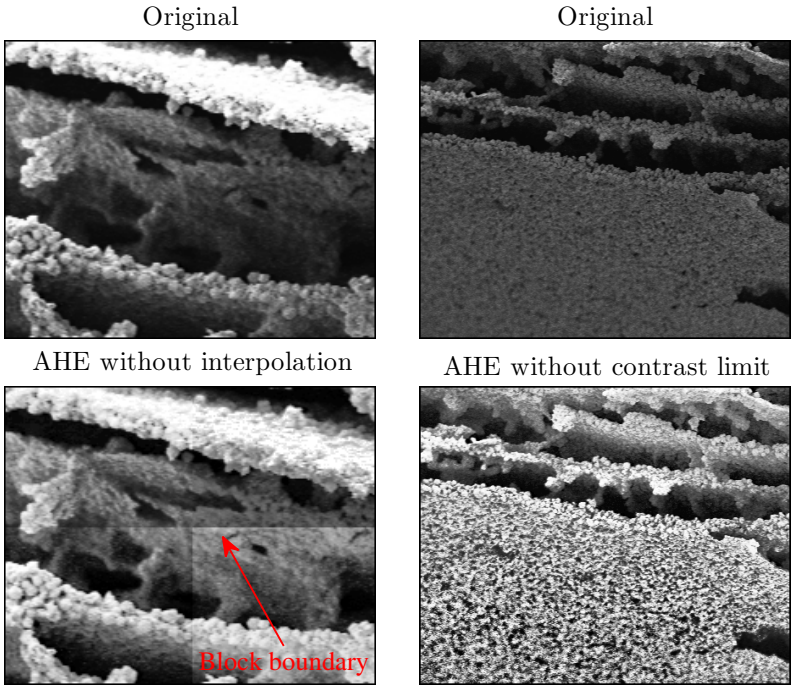

Figure 3.2: Problems related to processing images with AHE. In the top row are the original image patches and below them are the results of block-wise histogram equalization.

problematic results of AHE processing are shown below them. The bottom left image shows the block boundary lines that are clearly visible and disturbing. The bottom right image shows that AHE processing a homogeneous region leads to unnaturally pronounced texture pattern.

3.2.1 Interpolation

One way to remove visible transition lines in AHE processed images is to apply interpolation at block boundaries. As each block is transformed using different transform functions, similar input gray levels may be transformed into more dissimilar output gray levels in adjacent blocks. If the transformed pixels are close to a block boundary, the difference is visible in the result image as unnatural transition lines.

These aberrations can be removed effectively using interpolation where the pixels near block boundaries obtain their new values based on the distance to the neighboring blocks. The method has the same principle as bilinear interpolation, where the weights are inversely proportional to the distance from the interpolated pixel.

First a region $R_{M\times N}$ is selected containing all the pixels that need interpolation, and the new interpolated values are denoted by $S_{M\times N}$. The position of R is chosen to have its corners in the centers of the surrounding blocks. An illustration of this is shown in the Figure 3.3, where R is visualized by a dashed line and the block

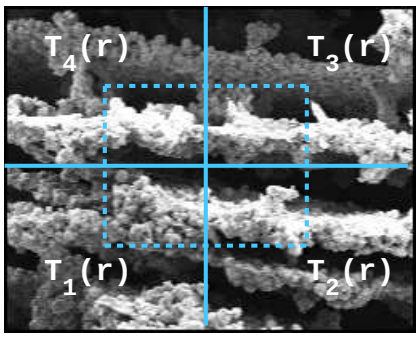

Figure 3.3: Interpolation of adaptive histogram equalized images. The region of interpolation R is visualized with dashed cyan line. The solid lines indicate blocks which have been independently processed with histogram equalization using transform function $T_k(r)$.

boundaries are drawn in continuous lines. In the figure, $T_k(r)$ is the transform function of the underlying block. Once region R is selected, it is transformed using the four different transform functions from the surrounding blocks resulting in four histogram equalized images S_k where $k \in \{1, 2, 3, 4\}$. The interpolated region S is then acquired by weighted averaging of S_k where the weights are proportional to the distance from the result image pixel $S(i, j)$ to the centers of the surrounding blocks.

In equation form, region S is written as

$$\begin{aligned} S(i,j) = &\frac{i(N-j)}{MN} S_1(i,j) + \frac{ij}{MN} S_2(i,j) \\ &+ \frac{(M-i)j}{MN} S_3(i,j) + \frac{(M-i)(N-j)}{MN} S_4(i,j) \end{aligned} \tag{3.4}$$

where $i \in \{1, 2, \ldots, M\}$ and $j \in \{1, 2, \ldots, N\}$. For example, the upper left corner of S, *i.e.*, $S(1,1)$ is the closest position to the center of upper left block. Therefore, $S_4(1,1)$ is given weight close to 1 and the other transformed pixels $S_1(1,1)$, $S_2(1,1)$ and $S_3(1,1)$ are given weights close to 0. Each pixel within R is then processed using the described method. This interpolation method completely removes visible lines at the block boundaries as shown in Figure 3.4. The process is repeated for each

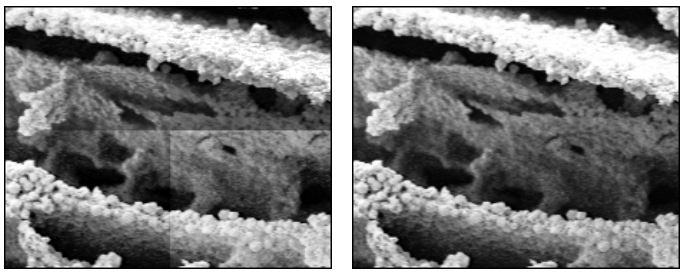

Figure 3.4: A comparison between AHE processed images with and without interpolation. The right image has been interpolated at block boundaries and shows that the boundaries are completely removed.

block boundary in the image producing smooth block transitions. This interpolation method is implemented in MATLAB as the function `adapthisteq`.

3.2.2 Contrast Limiting

Sometimes AHE produces images that have excessive amount of contrast enhancement due to homogeneous regions in the input image. As homogeneous regions induce high peaks in the histogram, and high peaks cause high slopes in the transform function, contrast enhancement for such regions becomes unnecessarily high.

The problem is solved by limiting the maximum amount of contrast enhancement a transform function can apply. This method is known as Contrast limited histogram equalization (CLAHE). It operates by setting a constraint for the slope of the transform function that is equivalent to setting a limit for the height of the histogram. The height is limited by defining a clipping level above which values in the histogram are clipped.

Merely clipping and discarding values above the clipping level causes the transform function to lack mapping for some gray levels [15]. It can be avoided by

redistributing the clipped area evenly to the bottom of the histogram as depicted in Figure 3.5. The clipping level is denoted by a dashed red line and the redistributed

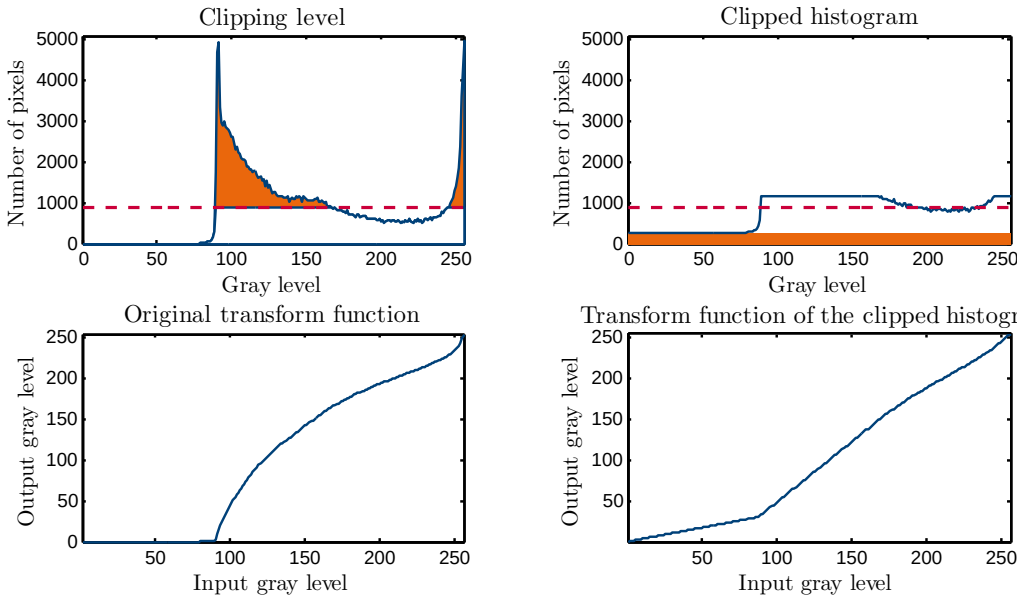

Figure 3.5: Contrast limited adaptive histogram equalization. The top row shows the original histogram and the clipped histogram where the clipping level is drawn in red dashed line. The bottom row shows the corresponding transform functions. In the left histogram, the orange area above the clipping level is evenly distributed among all gray levels in the right image. The effect of clipping is seen from the transform functions where the maximum slope of the graph is smaller in the right transform function.

area by orange color. Below the histogram graphs are the corresponding transform functions that show the effect of clipping. The images visualize that clipping and redistributing causes the histogram to rise above the original clipping limit, which might not be desired. If the true clipping limit is needed, an algorithm for computing it is provided in [15].

A slightly different variation of redistribution is to distribute excess pixels only to bins below the clipping level, which ensures that the modified histogram will not rise above the original clipping level. The MATLAB implementation `adapthisteq` uses this method. A comparison between images processed by AHE and CLAHE is presented in Figure 3.6, where the right image shows the effect of clipping histogram at level 0.01. The level is specified in normalized values such that the sum of histogram bins equals 1. In the left image, contrast enhancement is unrestricted.

One way to explain the outcome of redistribution of pixels is that the clipping level determines how close to the average of the histogram each bin will become, *i.e.*, how uniform the histogram will become. Selecting a clipping level high enough results in no clipping and unrestricted contrast enhancement whereas choosing a clipping level low enough, so that every histogram bin is clipped, leads to identity transform

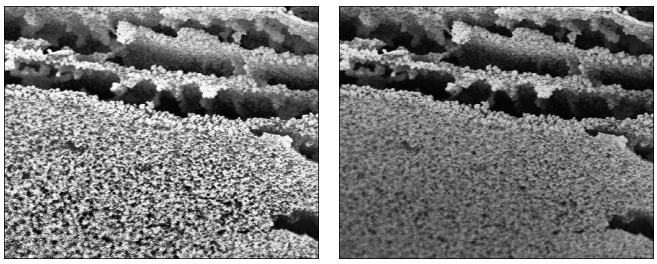

Figure 3.6: A comparison between AHE and CLAHE processed images. Both of the images have been interpolated at block boundaries. The left image shows the result of unrestricted contrast enhancement and the right shows the result of using clipping level of 0.01. The value is specified in normalized values where the sum of the histogram bins equals one.

function and the transformed image stays identical to the original one. An example image of CLAHE processed image using MATLAB's inbuilt function `adapthisteq` is presented in Figure 3.7. Both histograms below the processed images are flat but in the right CLAHE processed image also the histograms of each block are flat. This

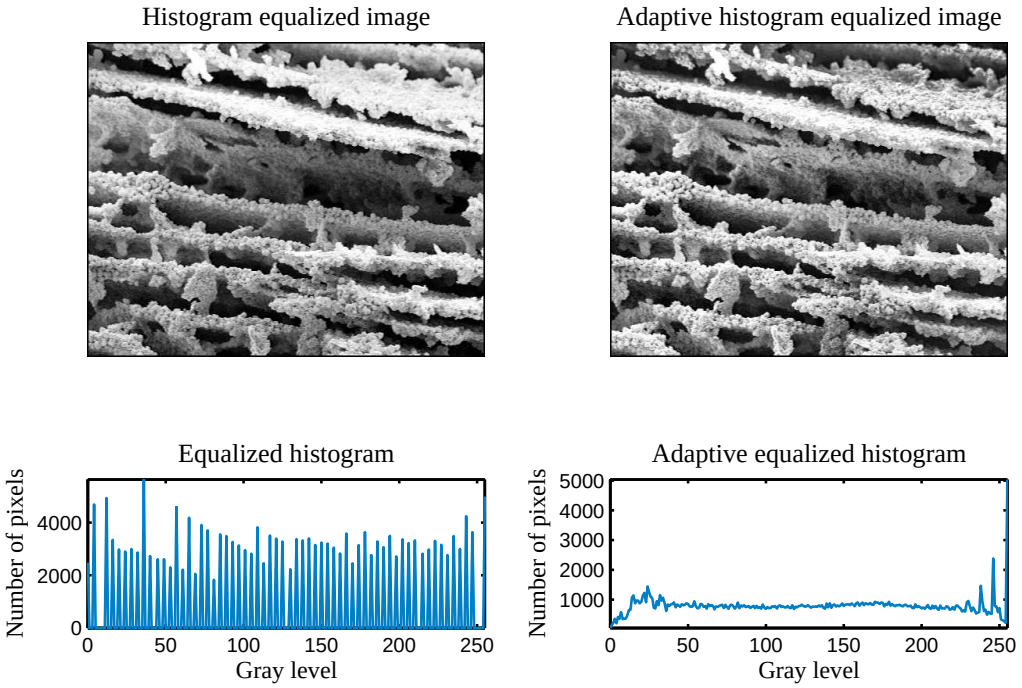

Figure 3.7: A comparison between HE and CLAHE. In the top-left corner is histogram equalized image. In the top-right corner is CLAHE processed image. Below these images are the corresponding histograms.

is seen as improved local contrast. For example, by comparing the top right corners of the images where there is a bright material region, more detail is visible in the CLAHE processed image.

3.3 Morphological Operations

Mathematical morphology is a branch in digital image processing that is traditionally used to analyze and process objects in images based on their shape rather than numerical values. It has the ability to simplify image data while preserving the most important characteristics [16]. Mathematical morphology is applicable to a wide range of image processing problems such as shape detection, noise filtering, image enhancement, segmentation and simplification. Moreover, it can be used for binary images, grayscale images and color images, which makes it even more versatile. [17]

Morphological operations operate via structuring element that probes the image for a specific condition. In the simplest case, the condition is tested with pixel-wise operations between the image and the structuring element. Depending on the outcome of the test, a value is written to the output image at the position indicated by the origin of the structuring element. Two examples of structuring elements are visualized in Figure 3.8 where the origin is marked in boldface text.

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & \mathbf{1} & 0 \\ 1 & 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & \mathbf{1} & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Figure 3.8: Examples of structuring elements. The origin, where the output value is written in the result image, is visualized in boldface text.

Mathematical morphology includes several operations that all are divisible into smaller set operations where one of the most primitive ones is translation. It is expressed as

$$A_{\mathbf{z}} = \{\mathbf{a} + \mathbf{z} \mid \mathbf{a} \in A\} \tag{3.5}$$

where $A \subset \mathbb{Z}^2$ is a set representing binary image and a subset of two dimensional integer space. Each element in the set is a vector holding the x and y coordinates of a white pixel in the image. Here $\mathbf{z} \in \mathbb{Z}^2$ is also a two dimensional vector for translating the elements in A. Translation simply moves every pixel of A by amount determined by $\mathbf{z}$.

Another basic operation, which forms the basis of more complex operations, is reflection. It is defined as

$$\hat{A} = \{-\mathbf{a} \mid \mathbf{a} \in A\}\,. \tag{3.6}$$

Reflecting an image equals to taking the opposite of each element in the image. In effect, $\hat{A}$ is a reflection of A about origin. [14]

3.3.1 Dilation

Dilation is a basic morphological operation which is mathematically expressed as

$$A \oplus B = \left\{ \mathbf{z} \mid (\hat{B})_{\mathbf{z}} \cap A \neq \emptyset \right\}, \tag{3.7}$$

where A is the binary image and B is the structuring element. Here B is also a set of x and y coordinates specifying white pixels in the structuring element. Dilation of image A by structuring element B is the set of all translations of $\hat{B}$ that have nonempty intersection with A. In practice, this means expanding regions in the image according to the shape and size of the structuring element. One way of visualizing the operation is to consider a field of seeds where each white pixel in A grows its own structuring element. The union of these structuring elements is then equivalent to the dilated image.

For calculation of dilated image, the structuring element is moved through every possible position in the image and the following condition is tested at each position. If the intersection between the image and the structuring element is not empty, 1 is written to the position of origin in the result image.

Grayscale morphology is similar to binary morphology, but the output value may vary within the same range as the image. In grayscale morphology, it is also possible for the structuring element to have varying intensities, in which case the structuring element is *nonflat*. However, in this work, only *flat* structuring elements are considered, *i.e.*, each member of the structuring element has equal weight. For example, the previously introduced structuring elements are flat.

Dilation of grayscale image f by a flat structuring element B is written as

$$[f \oplus B](x, y) = \max_{(s,t) \in B} \{f(x + s, y + t)\}. \tag{3.8}$$

It operates by moving the structuring element through each possible location in the image. The output value at each location is the maximum value among the values indicated by the structuring element, and it is written to the position of origin in the output image. Grayscale dilation expands bright features in the image and removes dark features that are smaller than the structuring element. [14]

Two examples of this operation are depicted in Figure 3.9 where the image in the top row is a binary image and the image below it is a grayscale image. The used structuring element has been drawn in red color over the original images in the upper left corner. As the image shows, the borders of the letter 'a' have expanded according to the geometry of the structuring element. For example, the amount of expansion in upward direction is determined by the distance from the origin of the structuring element to the uppermost coordinate of the structuring element. In the

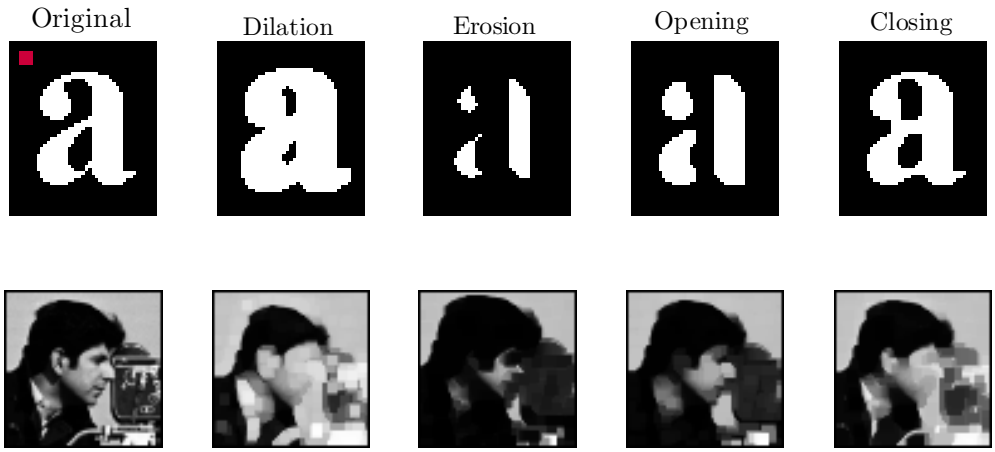

Figure 3.9: Morphological operations. The top row shows binary morphological operations and the bottom row shows the corresponding grayscale operations. The used structuring elements have been visualized in the upper-left corners of the original images.

case of square shaped structuring element, the directions which receive the most expansion are upper-right, upper-left, lower-right and lower-left since the distance to the origin is the longest.

Various different structuring elements exist and some of the most commonly used are diamond, line, square, ball and disk. Because the outcome of each morphological operation depends heavily on the shape and size of the structuring element, it is chosen according to the task.

3.3.2 Erosion

Erosion is another basic operation in mathematical morphology. It is the dual operation of dilation as it shrinks regions in an image. Mathematically it is written as

$$A \ominus B = \{z \mid B_z \subseteq A\}, \tag{3.9}$$

where A is the image and B is the structuring element. Binary erosion is the set of all translation vectors that translate structuring element B so that it is a subset of A.

Where dilation expands regions in a binary image, erosion shrinks them. To compute erosion for a binary image structuring element is moved through every possible position in the image and the following condition is tested at each position. If all the ones in the structuring element are in a position where there is one also in the image, an output of 1 is written to the position of origin in the output image. In other words, if the structuring element is completely within white regions of the binary image, an output of 1 is written to the position of origin. In any other case,

an output of 0 is written .

Grayscale erosion is otherwise the same as grayscale dilation but instead of selecting the maximum value, minimum is taken. For a flat structuring element B, it is defined as

$$[f \ominus B](x, y) = \min_{(s,t)\in B}\{f(x + s, y + t)\}. \tag{3.10}$$

Erosion is minimum filtering where the size of the window is determined by the structuring element. Computing the erosion of an image is done in same manner as in grayscale dilation. Images processed by grayscale erosion have their dark features expanded and the smallest light features removed. Examples of both binary and grayscale erosion are displayed in Figure 3.9.

3.3.3 Opening

Opening of a binary image A with structuring element B is defined by

$$A \circ B = (A \ominus B) \oplus B \tag{3.11}$$

where $\oplus$ and $\ominus$ denotes dilation and erosion respectively. This means simply processing the image consecutively with erosion and dilation. Binary opening removes certain image details smaller than the structuring element without distorting features that remain unsuppressed. For example, using a disk shaped structuring element removes ridges and islands smaller than the structuring element while smoothing uneven contours. Therefore, it may be used to remove noise in binary images.

Grayscale opening is almost the same binary opening and is expressed as

$$f \circ B = (f \ominus B) \oplus B \tag{3.12}$$

where the image f is now a grayscale image. It removes light features and preserves dark features that are smaller than the structuring element without distorting the most prominent features in the image. Figure 3.9 shows how opening simplifies the image by shifting the values of small light features towards darkest values in the local neighborhood. [16; 17]

3.3.4 Closing

Closing is the counterpart of opening, in which the only difference is the order of operations. Closing of image A with structuring element B is defined by

$$A \bullet B = (A \oplus B) \ominus B \tag{3.13}$$

where $\oplus$ is dilation and $\ominus$ is erosion. Binary closing fills gaps, removes small holes and inward bumps on contours without distorting features that remain unsuppressed, which is the opposite of opening.

Similar to the binary counterpart, the definition of grayscale closing is

$$f \bullet B = (f \oplus B) \ominus B, \tag{3.14}$$

which preserves light features and removes dark features that are smaller than the structuring element. Figure 3.9 demonstrates how closing simplifies the image by shifting the values of small dark features towards brightest values according to the structuring element. [16; 17]

3.3.5 Top Hat Transform

Top-hat transform, also known as white top-hat transform, is used to extract bright objects from their background. The definition of top-hat transform is

$$TH_{\mathrm{B}}(f) = f - (f \circ B) \tag{3.15}$$

where f is a grayscale image and B is a structuring element. In other words, top-hat transform is the opening of an image that is then subtracted from the original image. It preserves features from a grayscale image that are removed in morphological opening. As opening removes bright objects that cannot contain the structuring element, they become present in the top-hat transformed image.

A basic example of its use is to remove non-uniformly illuminated background from an image. In such case, the structuring element is chosen to be larger than the foreground objects, which causes them to be removed in the opening while leaving background intact. In the next step, subtracting original image from the opened one restores foreground objects but flattens the background. [14]

An example of top-hat transform is shown in Figure 3.10. The structuring element used in the transforms is visualized in the top-left corner of the original image. As it is seen from the image, top-hat transform highlights bright areas that fit within the structuring element. For example, the photographers ear is highlighted because erosion completely removes it, which is why it is not expanded in dilation. Lastly, subtracting the dilated image from the original one highlights it.

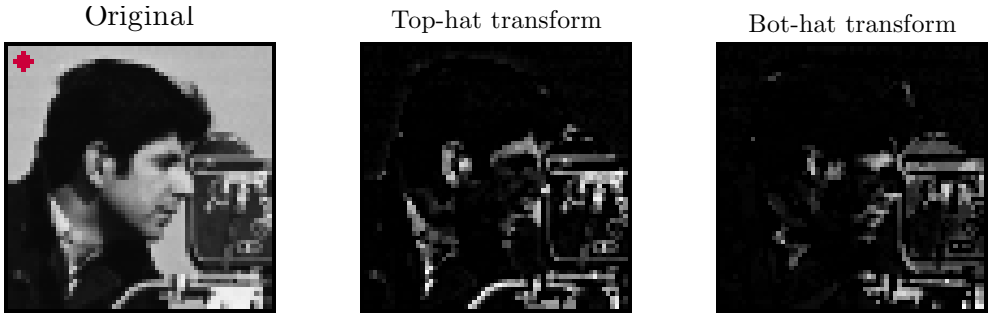

Figure 3.10: Morphological operations. The structuring element that was used in the transforms is shown in red color in the upper-left corner of the original image.

3.3.6 Bottom Hat Transform

Bottom-hat transform, also known as black top-hat transform, is the opposite of top-hat transform. It is written as

$$BH_{\mathrm{B}}(f) = (f \bullet B) - f. \tag{3.16}$$

Bottom-hat transform is closing of an image from which the original image is subtracted. It is very similar to top-hat transform, but instead of finding bright features, it finds dark features.

Bottom-hat transform is useful for finding gaps and holes that are smaller than the structuring element. An example of bottom-hat transformed image is depicted in Figure 3.10 where the eye of the photographer is highlighted. This is because the brighter intensities of the surrounding skin replaces the eye area during grayscale dilation. And since the eye is removed, it was not recovered in erosion. This leads to high difference between the original image and the closed image.

Mathematical morphology extends much further than the basics described in this chapter. Even some of the operations used in this work, such as connected component labeling and morphological reconstruction, were omitted due to the vast number existing methods. For further reading on the topic, see for example [18; 19].

3.4 Supervised Learning

Supervised learning is an approach where labeled samples are shown to an algorithm which tries to learn the relationship between the samples and the labels. It is typically applied to *classification* problems where the label of a previously unseen sample is predicted. [20]

Classification is simply explained with the following example. Suppose images of fish where each image contains either a salmon or a sea bass. The goal of classifi-

cation is to automatically divide these images into groups, also known as *categories* or *classes*, where one group contains only images of salmon and the other contains only images of sea bass. For humans it is simple task to divide the images into these groups because after seeing images of salmon and sea bass we notice differences in the *features*, *e.g.*, length and brightness of the fish that we use in the discrimination.

Similarly in automatic classification, we can first extract features from images of salmons and sea bass, for example, measure the average brightness and length of the fish and pass this information to a classifier. The classifier then tries to learn the general pattern of these features for each class. The image set of N images used in training the classifier is called *training data* where the images are denoted by X_i, $i \in \{1, \ldots, N\}$ and the corresponding class labels by $c_i \in \{0, 1\}$. For simplicity, salmon and sea bass classes have been replaced with values 0 and 1 respectively. Now, for a previously unseen image X to be classified, features are extracted and shown to the classifier which then compares the features to the learned patterns of both classes. The class having the best correspondence is chosen as the classification result. [11; 21]

A desired property of a classifier is *generalization*, *i.e.*, the ability to classify correctly new images that were not used in training. Generalization is important because the training data usually contains only a tiny fraction of all possible samples. A classifier's ability to generalize can be objectively measured using *test data*, which is a distinct set of images X_i $i \in \{N + 1, \ldots, N + M\}$ that has not been shown to the classifier. For the test images X_i, the corresponding labels c_i are also known. A common measure to evaluate classification performance is *accuracy*, which is the proportion of correctly classified samples in the test set. Evaluating classifier's ability to generalize with the training images leads to optimistic results. This is because the classifier has been trained using a small portion of all the possible samples and might have learned patters that characterize the training data rather than the classes in general. A classifier that classifies well training images but not as well new images is said to *overfit*. [20; 11]

Many classifiers exist where the differences between them lie in linearity, interpretability and speed. Simplicity and interpretability are the main advantages of linear classifiers, of which common examples are logistic regression, linear support vector machine, and naive Bayes. They can outperform nonlinear methods in cases when the amount of training data is small, the signal-to-noise-ratio is low or the data is sparse. On the other hand, nonlinear classifiers are capable of finding more complex structures from the data but are generally more difficult to interpret. Examples of nonlinear classifiers are multilayer neural networks and nonlinear support vector machine. From now on, the focus will be on linear classification due to simplicity and interpretability. [22; 23]

For the classification problems in this thesis, logistic regression is chosen due to simplicity, probabilistic output and sparsity promoting properties via regularization. It is a statistical classification method for predicting the class of a feature vector. The model was created for the purpose of using means of linear regression for categorical data. Therefore, it is regarded as a tool for classification rather than regression. Logistic regression can be applied to multiclass problems, but in this thesis only binary cases are discussed.

3.4.1 Logistic Regression Model

Suppose the feature vector $\mathbf{x} \in \mathbb{R}^p$ belongs to class $c \in \{0, 1\}$. Logistic regression models the posterior probability $p(C = c \mid \mathbf{x})$ of the feature vector $\mathbf{x}$ to belong to the class c. The model is defined as

$$\text{logit}\left(p(C = c \mid \mathbf{x})\right) = \log\left(\frac{p(C = c \mid \mathbf{x})}{1 - p(C = c \mid \mathbf{x})}\right) = \beta_0 + \boldsymbol{\beta}^T\mathbf{x} \tag{3.17}$$

where β_0 and $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_p]^T$ are the model parameters jointly denoted by $\boldsymbol{\theta} = [\beta_0, \boldsymbol{\beta}^T]^T$. Instead of modeling the conditional probabilities directly using a linear function of $\mathbf{x}$, logit transform is applied first. The transform is needed because probabilities vary within range $p(C = c \mid \mathbf{x}) \in [0, 1]$ where the linear predictor $\beta_0 + \boldsymbol{\beta}^T\mathbf{x}$ is not restricted and may take any real value. Thus logit transform may be seen as a mapping from the range $[0, 1]$ to the entire real axis after which linear modeling becomes more reasonable. Additionally, it is assumed that the probabilities become linear after the transform. A graph of the logit function is presented in Figure 3.11.

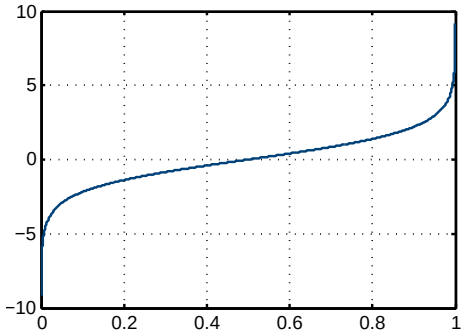

Figure 3.11: Logit function

By solving the equation (3.17) for $p(C = c \mid \mathbf{x})$, the expression

$$p(C = c \mid \mathbf{x}) = \frac{\exp\left(\beta_0 + \boldsymbol{\beta}^T\mathbf{x}\right)}{1 + \exp\left(\beta_0 + \boldsymbol{\beta}^T\mathbf{x}\right)} = \frac{1}{1 + \exp\left(-(\beta_0 + \boldsymbol{\beta}^T\mathbf{x})\right)} \tag{3.18}$$

is obtained, which gives the posterior probability of class c given feature vector $\mathbf{x}$ and the model parameters β_0 and $\boldsymbol{\beta}$. The output of the linear predictor $\beta_0 + \boldsymbol{\beta}^T\mathbf{x}$ is given to sigmoid function that is the inverse of logit function. In the case of two classes, only one set of coefficients β_0 and $\boldsymbol{\beta}$ are needed since we may define that $p(C = 0 \mid \mathbf{x}) = 1 - p(C = 1 \mid \mathbf{x})$, which means that only one linear function is used to predict the class probabilities for both classes.

Logistic regression is a linear classifier even though a nonlinear transform is applied to the linear predictor because it produces linear decision boundaries. This is observed by thresholding the predicted probabilities by t after which the decision boundary becomes the set of feature vectors $\{\mathbf{x} \mid \beta_0 + \boldsymbol{\beta}^T\mathbf{x} = \text{logit}(t)\}$. A typical choice for the threshold level is $t = 0.5$ since it specifies a point where both classes are equally probable. This choice means that when the linear predictor produces a positive value, one of the classes is more probable whereas a negative value suggests higher probability for the other class. [22]

3.4.2 Learning the Model Parameters

The model parameters for logistic regression are learned by maximum likelihood estimation, which is a common parameter estimation method in statistics. It is often simpler than alternative methods and has desired convergence properties as the number of samples increase [11].

Suppose we have feature vectors $\mathbf{x}_i$, $i \in \{1, \ldots, N\}$ and the corresponding true class labels $c_i \in \{0, 1\}$ as training data. For fixed model parameters $\boldsymbol{\theta}$, the probability of observing the training samples $\mathbf{x}_i$ is

$$L(\boldsymbol{\theta}, \mathbf{x}_1, \ldots, \mathbf{x}_N) = p(\mathbf{x}_1, \ldots, \mathbf{x}_N \mid \boldsymbol{\theta}) = \prod_{i=1}^{N} P(C = c_i \mid \mathbf{x}_i), \tag{3.19}$$

which gives the likelihood of the training data. In maximum likelihood estimation, the above is considered as a function of the model parameters $\boldsymbol{\theta}$ while training samples are kept fixed. Solving it for the function maximizing model parameters yields the maximum likelihood estimate $\hat{\boldsymbol{\theta}}$.

Maximizing the logarithm of likelihood function (log-likelihood) gives the same solution as maximizing the likelihood function itself because logarithm is a monotonic function. Working with log-likelihood is considered more convenient for analytical purposes and the risk of arithmetic underflow is smaller. [11; 20]

For simplicity, let $P(C = 1 \mid \mathbf{x}_i, \boldsymbol{\theta}) = p(\mathbf{x}_i)$ and $P(C = 0 \mid \mathbf{x}_i, \boldsymbol{\theta}) = 1 - p(\mathbf{x}_i)$.

Now the objective function is written, for two classes, as

$$l(\boldsymbol{\theta}) = \log\left(L(\boldsymbol{\theta}, \mathbf{x}_1, \ldots, \mathbf{x}_N)\right) = \sum_{i=1}^{N} \left[c_i \log(p(\mathbf{x}_i)) + (1 - c_i) \log(1 - p(\mathbf{x}_i))\right] \quad (3.20)$$

where N is the number of samples and p is the length of the parameter vector $\boldsymbol{\beta}$. The problem with this parameter estimation method is in the case of linearly separable classes in the feature space. In such case, there exist multiple plausible parameter vectors because any solution separating the classes minimizes the objective function resulting in a parameter vector of infinite magnitude [20]. It is clear that this saturation problem greatly reduces the classifiers ability to generalize.

One way to prevent this problem is to introduce *regularization* term in the objective function. Regularized log-likelihood is expressed as

$$\begin{aligned} l(\boldsymbol{\theta}) &= \sum_{i=1}^{N} \left[c_i \log(p(\mathbf{x}_i)) + (1 - c_i) \log(1 - p(\mathbf{x}_i))\right] \\ &\qquad \text{subject to } \|\boldsymbol{\beta}\|_1 < t \end{aligned} \quad (3.21)$$

or equivalently

$$l(\boldsymbol{\theta}) = \sum_{i=1}^{N} \left[c_i \log(p(\mathbf{x}_i)) + (1 - c_i) \log(1 - p(\mathbf{x}_i))\right] - \lambda \sum_{j=1}^{p} |\beta_j| \quad (3.22)$$

where λ (or t) is a parameter that constrains the magnitude of L_1-norm of the parameter vector $\boldsymbol{\beta}$. For every value of λ, there exists one corresponding value for t. Regularization has the effect of favoring some parameters over others based on how the penalty is formulated.

Constraining the L_1-norm of the parameter vector is called the LASSO (Least Absolute Shrinkage and Selection Operator), which has an interesting property of providing sparse solutions where some coefficients in the parameter vector $\boldsymbol{\beta}$ are zero. Varying the parameter λ determines the sparsity of the solution. Having some of the coefficients zero means that not all features are used in predicting the class probabilities and thus can be left out from feature extraction giving faster computation times. [24]

Studies have shown that estimates obtained using LASSO regularization are insensitive to high correlation in the features [24; 25]. This is a useful property in many real world applications when it is difficult to choose features that have no mutual correlation. A typical method to remove correlation between features involves Principal component analysis (PCA), which is used to transform features into a coordinate system where no such correlations are present. Since LASSO is

able to manage correlating features, there is no need for PCA in this regard leading to simpler and faster classification procedure.

One problem in LASSO regularization arises when $p > N$, *i.e.*, the number of features is higher than the number of samples. Under these circumstances, LASSO regularized estimates may have at most N non-zero coefficients, regardless of the lambda value, resulting in heavy feature selectivity. Other regularization methods have been developed to overcome some of the shortcomings of LASSO. One such method is the elastic net penalty, which is a mixture of L_1 (LASSO) and L_2 (Ridge) penalties. It can produce parameter estimates with more than N nonzero coefficients in the case when there are more features than samples. By varying the mixture parameter, one can control the amount of sparsity at different values of lambda. [22]

The parameters that maximize the objective function 3.22 for various different lambda values, also known as the regularization path, may be solved programmatically. One efficient implementation to do this is the Glmnet package available for MATLAB [26]. It uses cyclical coordinate descent algorithm where the optimization problem is reduced into multiple univariate problems where only one parameter is allowed to vary and others are kept fixed. Cycling through each coefficient one at a time until convergence provides the LASSO regularized estimate $\hat{\boldsymbol{\theta}}$ for one λ. Computation of the whole regularization path is fast since coefficients from previous iteration may be used as an initial guess for the successive iterations [22]. Geometrically cyclical coordinate descent algorithm moves in parameter space towards local minimum of the objective function by moving only along one coordinate dimension at a time.

3.4.3 LASSO Regularization Geometry

Some insight on the LASSO regularization can be obtained from geometrical interpretation. Figure 3.12 shows the contours of equivalent likelihood in the parameter space denoted by colors ranging from red to yellow, *i.e.*, the positions where the likelihood function gives the same value with various different parameters $\boldsymbol{\beta}$. Red contours correspond to higher likelihood while yellow denotes lower likelihood.

In the middle of the contours is the point $\hat{\boldsymbol{\beta}}$ where the unregularized log-likelihood function gives the maximum value (ML-estimate). The diamond shape around origin visualizes a region $\{\boldsymbol{\beta} \mid \|\boldsymbol{\beta}\|_1 \leq t\}$ corresponding to t (or λ). Parameters within this region are plausible solutions for the LASSO regularized objective function. The point $\hat{\boldsymbol{\beta}}_{LASSO}$ is the solution that maximizes the LASSO regularized objective function having lambda set to λ.

In this simplified example, the contours of the objective function are round, making the regularized solution equal to the Euclidean projection of $\hat{\boldsymbol{\beta}}$ into the diamond shaped set $\{\boldsymbol{\beta} \mid \|\boldsymbol{\beta}\|_1 \leq t\}$. Euclidean projection means finding a member of the set

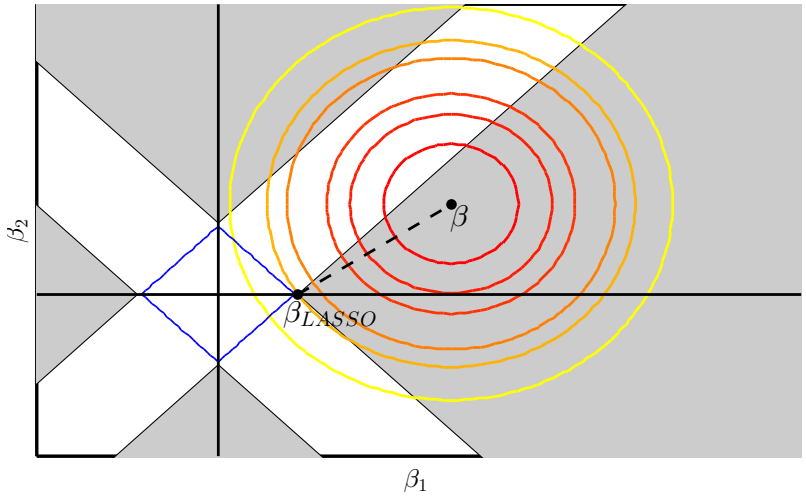

Figure 3.12: An illustration of LASSO regularization in the case of circular contours of equivalent likelihood. The unregularized parameter estimate is denoted by $\hat{\boldsymbol{\beta}}$ and the corresponding LASSO regularized estimate is $\hat{\boldsymbol{\beta}}_{LASSO}$. In parameter space, LASSO estimate is the Euclidean projection of the unregularized estimate to the diamond shaped set. The dashed line visualizes the projection.

that is closest to the point being projected on the set. The projection is visualized by a dashed line in the figure.

Gray areas in the image show regions where the Euclidean projection hits a corner of the diamond, thus resulting in a sparse solution. Now consider increasing the λ value, which is equivalent to decreasing t value. As t decreases, the size of the diamond reduces and the proportion of gray areas to white areas increases, increasing also the probability of sparse solution. Moreover, adding dimensions to the feature space further increases the ratio of gray areas to white areas, or more accurately the ratio of volumes, that leads more probably to a sparse solution.

A visualization of a case when the contours of equivalent likelihood are not round is shown in Figure 3.13. The effect of increasing the penalty is depicted for three different λ values corresponding to three diamond shapes centered at origin. The edge of the largest diamond intersects with one of the contours in a position where both of the parameters are non-zero. Same is true for the second largest diamond. The smallest diamond is small enough to have its intersection point at a corner resulting in a sparse solution.

3.5 Watershed Segmentation

Various types of approaches have been developed for image segmentation problems over the years. Examples of these include thresholding methods [27; 7], feature detection based methods such as edge detection together with edge linking [28; 29], region based methods [30; 31], clustering [32; 33], deformable model fitting [34] and

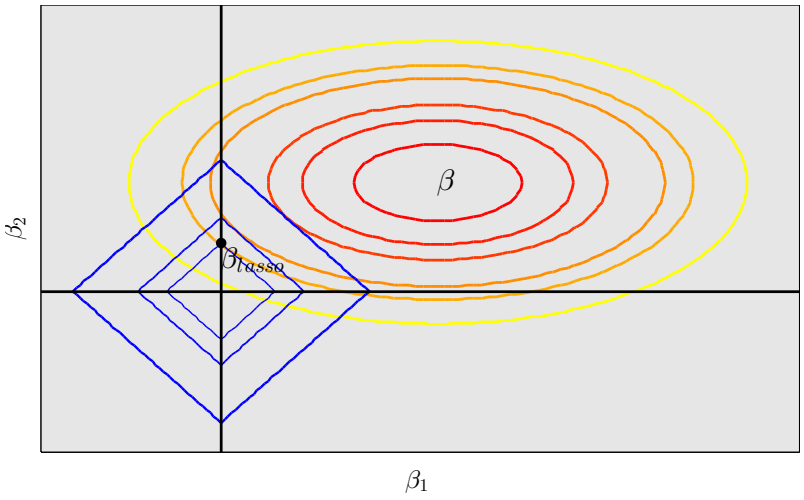

Figure 3.13: The effect of increasing regularization parameter λ. As the value of λ increases, the probability of a sparse solution grows. In this example, the highest λ gives a sparse solution as the contour of equivalent likelihood intersects with a corner of the diamond.

graph cuts [35]. In this work, a method called watershed segmentation is studied. It shares some concepts with the listed methods, *e.g.*, region growing, and is sometimes regarded as its own category of segmentation algorithms [14; 36] and sometimes not [37].

Watershed segmentation considers an image as a topographical relief where the intensity value of a pixel is viewed as height for that particular location. This relief is flooded by raising water level and allowing water into the relief only from positions of local minima in the image. As the water level rises, *catchment basins* begin to emerge and grow. When two of these catchment basins would merge into a bigger basin, a dam is built in between to prevent the merging. The water level keeps rising and dams are built until the relief is completely under water. At this point, only dams are above the water level and execution of the algorithm is stopped. The dams indicate the final segmentation result. Each catchment basin represents one segmented region separated from other regions by dams that are also the region boundaries.

Another description of the method is to imagine rain falling on the relief. The set of points from which water drops fall into the same local minimum, is a catchment basin. And the set of points from which the water drops would equally likely fall into two or more minima indicate the divide lines or watershed lines. [14]

It is common for object boundaries in images to have rapid intensity transitions. For this reason, watershed transform is regularly applied to gradient magnitude image, rather than the original image, to have the watershed boundaries form at the object boundaries [37]. This also causes low values in regions of homogeneous intensities where region boundary formation is less desired.

Traditional version of the method uses every local minima in the image as a water source. Due to the fact that natural images have many local minima, watershed algorithm has the tendency to produce highly oversegmented results [36; 14]. An example of this problem is visualized in Figure 3.14 where the upper left image is the original one, upper right is the gradient magnitude image obtained by 3×3 Sobel filtering and the bottom left image is the watershed segmentation result. The oversegmentation problem is clearly visible in the figure where each distinct region has been presented in random color. White lines between these regions denote the dams, *i.e.*, the watershed lines.

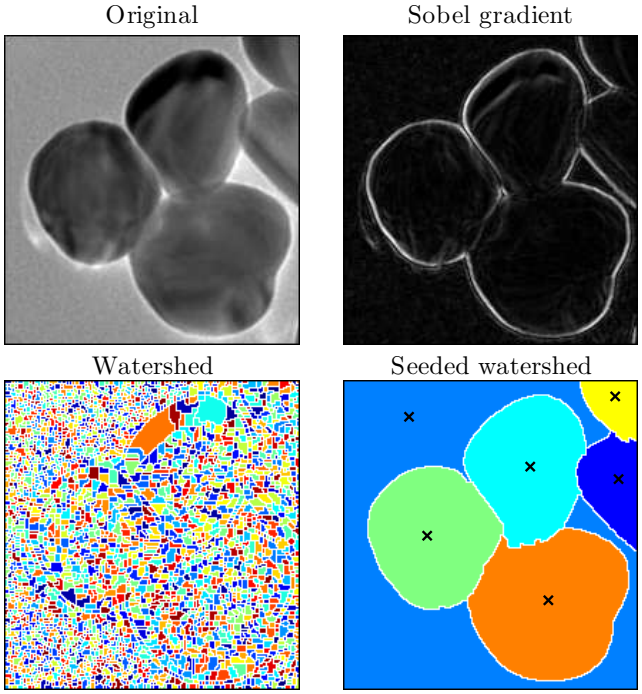

Figure 3.14: Watershed segmentation examples. Top left image is the original image and its convolution with Sobel operator is in the top right image. The bottom left image depicts unseeded watershed segmentation on the gradient image where every local minimum is used as a water source resulting in oversegmentation. The bottom right image shows seeded watershed segmentation on the same image, in which the manually selected seed points are indicated by crosses. The 3×3 Sobel operator is defined for x-direction as $S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ and $S_y = S_x^T$ for the y-direction. The gradient image, as in the top right plot, is acquired by $I_g(i,j) = \sqrt{G_x(i,j)^2 + G_y(i,j)^2}$, where G_x and G_y denote convolution of the original (upper left) image with the Sobel operator corresponding to the subscript direction.

In order to prevent the oversegmentation problem, a modified version has been developed to explicitly define the used water sources [38]. Water sources specified in such manner are called *seed points* or *markers*. The ability to select seed points

gives complete control over the number of regions in the end result, which fixes the oversegmentation problem. An example segmentation of seeded watershed method is presented in the bottom right image of Figure 3.14, where the same image is segmented using only manually selected seed points denoted by black crosses. It shows that the segmentation result is extremely dependent on the seed point selection because the number of used seed points is directly the number of segments in the end result.

The use of watershed segmentation embodies various benefits. It is able to produce continuous divide lines and contiguous regions since, by construction, the aim of the algorithm is to extract objects from the image. This property is particularly helpful when not all region boundaries are visible in the image. In addition, the watershed algorithm produces divide lines that correspond to natural characteristics in the image, and it is generally applicable to various types of images. Lastly, the user may affect the output of the method by specifying seed points. This is useful when it is desired to manually control the segmentation outcome or use some advanced automatic method to find the seed points. [36]

In this work, seeded watershed segmentation is applied for nanoparticle segmentation in case 2, where the seed points are found automatically from a distance transformed initial segmentation. This transform is next discussed in more detail.

3.6 Distance Transform

A method of wide application in image processing is distance transform, also known as distance map. It produces a grayscale output image containing distances from every pixel location to the nearest white pixel. Some examples of its use are computation of shape-descriptive features and object skeletons. [17]

For computation of the distance transform, various metrics can be used, *e.g.*, Euclidean distance, city-block distance or chessboard distance. An example of the transform is presented in the rightmost image of Figure 3.15 that has been computed in MATLAB using function `bwdist` and Euclidean distance. As the image shows, the distance to the nearest background pixel at the center of the particles is the highest, which is seen as the brightest values in the transformed image.

3.7 Receiver Operating Characteristic

Receiver operating characteristic (ROC) is a graph for visualizing and evaluating the performance of a classifier. In particular, the use of ROC graphs is beneficial when classes have imbalanced number of samples and when misclassification costs are uneven [21].

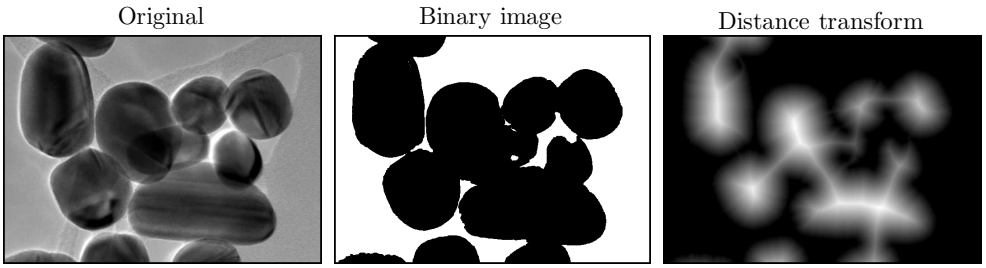

Figure 3.15: Distance transform. The images form left to right: Original image patch, thresholded binary image, distance transformed binary image. Each value in the distance transform describes a distance between two pixels in the binary image. The intensity of a pixel is the distance from its position to the nearest white pixel.

3.7.1 Confusion Matrix

In binary classification, the true class $y \in \{0, 1\}$ of a feature vector $\mathbf{x}$ is predicted where the prediction is denoted by $\hat{y} \in \{0, 1\}$. The outcome of the classification may be categorized into four different groups:

- *True positive*: Predicted class $\hat{y} = 1$ and true class $y = 1$
- *True negative*: Predicted class $\hat{y} = 0$ and true class $y = 0$
- *False positive*: Predicted class $\hat{y} = 1$ and true class $y = 0$
- *False negative*: Predicted class $\hat{y} = 0$ and true class $y = 1$.

Suppose some feature vectors are classified whose true classes are known. The instances in each group may be counted and presented in a table called *confusion matrix*. An example of it is presented in Table 3.1.

Table 3.1: Confusion matrix is a table of different classification outcomes. Suppose binary classification of a sample having true class y and the predicted class $\hat{y}$. The classification result may be categorized into one of the four classes. Confusion matrix is the basis of many performance metrics.

		True class y	
		1	**0**
Predicted class $\hat{y}$	**1**	**T**rue **P**ositives	**F**alse **P**ositives
	0	**F**alse **N**egatives	**T**rue **N**egatives
Column totals		**P**	**N**

Many different classification performance metrics are computed from the confusion matrix. Two such metrics are *True positive rate* (TPR) and *False positive rate*

(FPR). Since obtaining the true TPR and FPR of a classifier would require classification of every possible instance, these metrics are typically approximated from the test samples as

$$TPR = p(\hat{y} = 1 \mid y = 1) \approx \frac{TP}{P} \tag{3.23}$$

and

$$FPR = p(\hat{y} = 1 \mid y = 0) \approx \frac{FP}{N}. \tag{3.24}$$

Here the total number of positive samples P is the sum along the first column in confusion matrix, and total number of negative samples N is the sum along the second column. TPR, also known as *hit rate* and *recall*, is the ratio of successfully classified positive samples to all positive samples. Similarly, FPR is the ratio of incorrectly classified negative samples to all negative samples. [21]

3.7.2 ROC Graph

Once classification has been performed producing binary labels for some test data, the confusion matrix can be constructed. Based on the confusion matrix, TPR and FPR are plotted in two dimensional space where y-axis represents TPR and x-axis represents FPR. This is called receiver operating characteristic (ROC) space and graphs plotted in this space are called ROC graphs.

A discrete classifier is denoted by a single point in ROC space such as point A in Figure 3.16. Generally, the performance of a classifier is the better the closer to

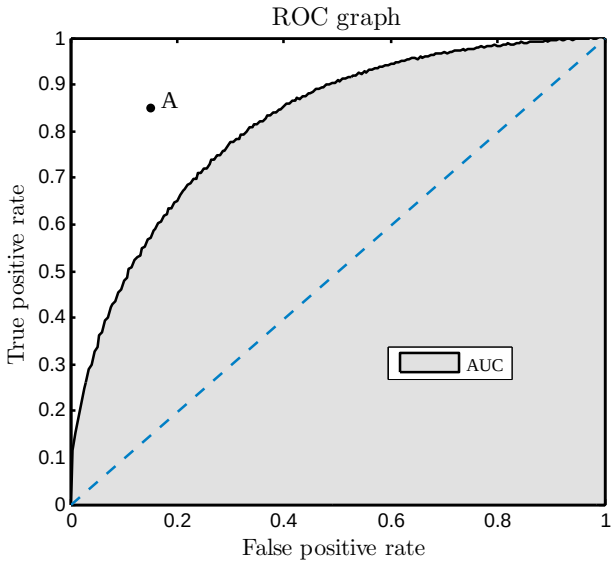

Figure 3.16: An ROC graph. A point in the graph denotes a classifier. A classifier is considered the better the closer it is to the upper left corner. Here, the dashed line corresponds to randomly guessing the classes for instances and no serious classifier should go below it because inverting every prediction would then lead to more accurate classification

the upper left corner it is in the figure. Such position indicates that all positive and negative samples were classified correctly. The dashed line corresponds to randomly guessing the class of a sample. For example, randomly guessing 50% of the test data to be positive samples yields the point $(0.5, 0.5)$ in ROC space because 50% of the positive and negative samples are classified correctly. By varying the ratio at which classes are randomly guessed, the dashed line is formed.

Classifiers such as naive Bayes, neural networks and logistic regression produce class scores or probabilities instead of actual class labels to indicate class membership. For these classifiers, the actual class labels are determined by thresholding. As an example, if the probability $p(y = 1 \mid X = \mathbf{x}) > 0.5$, it is reasonable to assume that the class is actually 1. However, depending on the problem domain, other than fixed threshold of 0.5 can lead to better results.

For these situations, ROC graph is a handy tool for evaluating the effect of different threshold levels. Having the threshold level vary $t \in [-\infty, \infty]$, and for every value calculate the corresponding TPR and FPR, a continuous line can be drawn in ROC space. An example image of a ROC graph is shown in Figure 3.16. Having a threshold level low enough assigns label $\hat{y} = 1$ to every sample, which corresponds to point $(1, 1)$ in ROC space because TPR = FPR = 1. Following the same logic, a high enough threshold results in point $(0, 0)$ in ROC space.

A common performance metric to evaluate classifiers is the *Area under curve* (AUC), which is drawn in gray in the figure. Statistically, it is the probability of a classifier to give a higher score or probability to a randomly chosen positive sample than to a randomly chosen negative sample. AUC is an interesting evaluation metric because it does not depend on the threshold level and it is invariant to class imbalance. Classes are imbalanced if the prior probabilities $p(y = 0) \neq p(y = 1)$. In practice, it means that a randomly chosen feature vector has higher probability to belong to one class than the other. [21]

4. CASE 1: PORE SEGMENTATION OF SEM IMAGES

Electron microscopy is an approach that allows precise analysis of materials and is considered a valuable tool in materials science. The images obtained in such way contain information about the imaged material that is relevant to researchers. In this chapter, the relevant information is the shape and size characteristics of pores in manufactured zeolite samples.

Calculating pore shape characteristics from zeolite images can be done manually from the images but is often time consuming. Additionally, automation allows processing of higher number of images within the same time window giving also more accurate measures of the pore shape characteristics. For these reasons, an automatic procedure for segmenting zeolite images is developed, tested and analyzed.

4.1 Data

The images in this case were acquired by scanning electron microscope (SEM), which is a valuable imaging method in analysis of nanoscale materials. It operates by emitting a focused beam of electrons at the surface of a specimen. The interaction between the specimen and the electron beam generate different types of signals, which are then measured and ordered to form the final image. [39]

Majority of the analyzed images in this case were provided by the Department of Materials and Environmental Chemistry from Stockholm University and the rest were from the Department of Materials Science of Tampere University of Technology. In total there are 18 SEM-images, each of which has the same magnification of $\times 200$ and dimensions 1280×1024.

The SEM images by itself do not contain any label information needed in training based methods. Therefore, it needs to be collected. The images were annotated using a script that was programmed specifically for this purpose in MATLAB. The annotation script produces binary image for each SEM image indicating locations of pores. The annotated images were then divided into two distinct sets, namely training set and test set. Out of 18 images, 10 was put in the training set and 8 in test set.

4.2 Objective

Image segmentation is a process where an image is divided into meaningful regions that typically hold image data of objects in the image. The aim is to create a method that analyzes an SEM image and creates a binary image of equal size that indicates the locations of pores in the SEM image. In this binary image, each pixel is assigned value 1 if there is a pore present in the corresponding position of the SEM image. Otherwise pixels take value 0. Since it is not possible for two pores to overlap, a simple binary image is sufficient to contain all the segmentation information.

The left image in Figure 4.1 shows an example SEM image of a zeolite sample. There are two types of porosity visible in the image: Big porosity and microporosity.

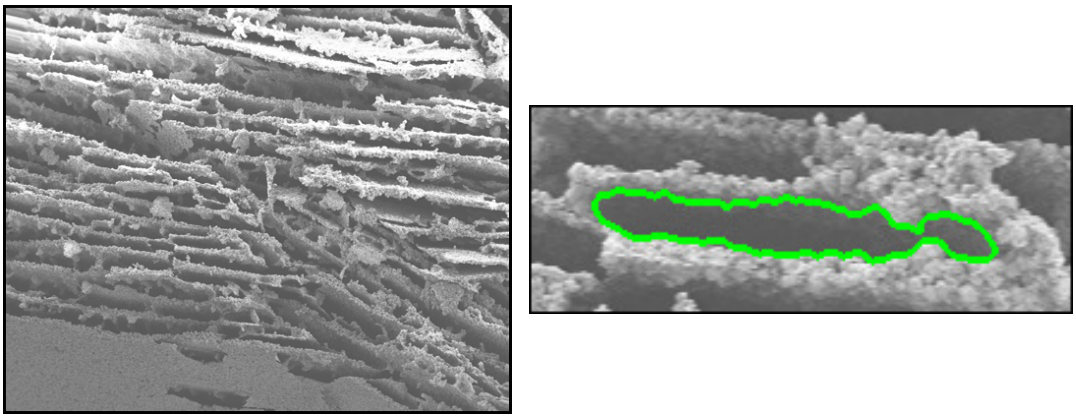

Figure 4.1: Scanning electron microscope image of zeolite. The left image is an unprocessed image directly from the microscope and the right image shows the perimeter of a pore.

All the holes visible in the left image are big pores. In the right image, a green line has been drawn to indicate the perimeter of a big pore. Microporosity is only visible in the right image, and it is recognized by grainy pattern constituting the walls. The dark holes between these grains are micropores. Because the shape of big pores has large impact on the overall strength of the structure, it is the main focus in this case. From this point forward, the term pore is used only to refer to big porosity.

The right image in Figure 4.1 illustrates also the general idea of how each pore should be segmented. However, there is no ground truth, making evaluation of the segmentation result somewhat problematic and subjective as each person would specify boundaries of pores into slightly different positions.

4.3 Implementation

A method based on the framework presented in Chapter 2 is studied for solving the pore segmentation problem. The main idea is to train a classifier that predicts the

class labels of each pixel. The classifier was chosen to be logistic regression mainly mainly due to speed, interpretability and sparsity promoting properties.

Figure 4.2 shows the block diagram of the proposed method. The upper part of the figure, separated by dashed gray line, shows the steps in the training phase. The lower part shows the actual segmentation phase. Blocks that are on the lighter colored background belong to preprocessing and blocks on the darker background are part of segmentation. The partitioning of the segmentation phase has the same structure as shown in the general block diagram in Figure 2.2. The arrow from

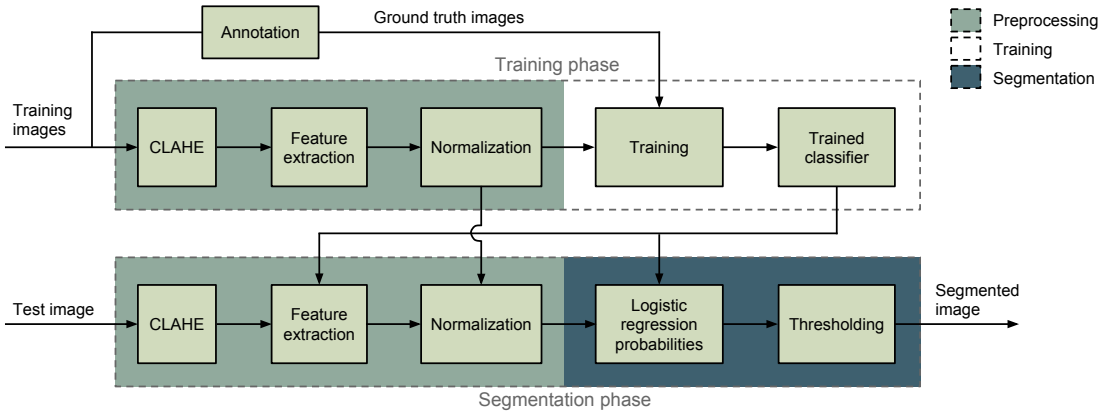

Figure 4.2: A block diagram of the case 1 segmentation pipeline. The areas on the lighter green background denote preprocessing and the darker blue-green color denotes the actual segmentation phase.

trained classifier to the feature extraction of segmentation phase denotes information of the sparsity of the classifier. As some of the model parameters are zero, the corresponding features need not to be computed. The other arrow between normalization steps denotes passing the normalization parameters used in training phase to the segmentation phase. In fact, this framework is implemented along the lines of [5], but instead of graph cutting, the probability image is thresholded. In addition, more advanced preprocessing is used here (CLAHE) than in the other framework.

Both the training phase and the segmentation phase begin with the same preprocessing steps as shown in the diagram. Next the preprocessing steps and their purpose in the framework is explained in more detail.

4.3.1 Preprocessing

The dimensions of the images are 1280×1024, which provides more than sufficient resolution for the task. Therefore, dimensions of the images are reduced 50% in order to decrease the time required for training without impacting significantly on the core information, *e.g.*, the shape and size of the pores.

By looking at the example SEM image in Figure 4.1, one can observe the lack of contrast in the lower parts of the image and higher intensity in the upper parts of the image. These problems could impair segmentation results if the chosen features are not invariant against them. Table 4.1 shows the pool of features used in the training phase. All of these features are affected by contrast variation meaning that

Table 4.1: The complete pool of features used in the training phase, from which the most important features are selected automatically.

Feature	Parameter	Values
local variance	window size	3,5,9,15,30,50,70
Sobel gradient	kernel size	3,5,9,15,30,50,70
morphological edges	disk radius	3,5,9,15,30,50,70
Gaussian low-pass	window size	3,5,9,15,30,50,70
morphological top-hat	disk radius	3,5,9,15,30,50,70
morphological bottom-hat	disk radius	3,5,9,15,30,50,70

the output of the feature is different between a region of high contrast and a region of poor contrast. This would have a negative impact on the classification result because class scores are linear combinations of features that have been monotonically transformed. This score is affected by contrast variation and therefore simple thresholding might prove problematic in determining the class labels.

The variation of contrast across the image is effectively normalized by processing the image using CLAHE. The parameter for CLAHE clipping level was chosen experimentally and the default (0.01) produced satisfactory results. This value is used to clip a histogram that has been normalized to sum up to 1. Therefore, every gray level containing more than 1% of the pixels in the image is clipped. The window size was chosen to be one sixth of the image dimensions, which allows some pore regions and some material regions to fit within the window. As a result, pore regions take lower values and non-pores higher values depending on the distribution of gray values withing the processing window, thus normalizing the contrast variation. A typical gain in classification accuracy given by CLAHE is 2 percentage points.

Next step in the preprocessing phase is feature extraction. Commonly, feature extraction is used to reduce the amount of data by preserving the most valuable information and discarding the rest to simplify the classification problem. However, in this case, the dimensionality of the data is drastically increased using redundant and correlated features. This approach is viable because LASSO regularization is robust against correlations in the features. Due to the feature selectivity, the best features are selected in the sense of maximum likelihood for logistic regression model.

The complete pool of features, for which the automatic selection is performed, is shown in Figure 4.1. The feature set was chosen to include various types of features using multiple window sizes in order to have the LASSO regularization choose the best features for the problem. In total, 42 feature images are extracted from one SEM image and are then vectorized and concatenated horizontally. All the feature matrices from the processed images are then concatenated vertically, resulting in one big feature matrix $\mathbf{X}$, in which one row contains the feature data from one pixel position in one of the processed images, and one column corresponds to feature data of a single feature extracted from all the processed images. Preprocessing the 10 training images yields the feature matrix $\mathbf{X}$ of size $3\ 235\ 840 \times 42$.

Once the feature matrix is complete, each feature, *i.e.*, column is normalized to have zero mean and unit standard deviation because otherwise the feature selectivity of LASSO becomes biased. This is explained through the following example. Suppose a high valued feature and a low valued feature that are equally good for a classification task but are scaled differently. Now a LASSO regularized classifier is trained with the two features and the corresponding coefficients are found. As the scaling is different for the two features, the lower valued feature needs a higher coefficient to have the same impact in prediction as the higher valued feature. For this reason, features that require smaller coefficient are selected more easily in the feature selection of the LASSO regularization, which penalizes L_1-norm of the coefficient vector.

This biased selection is prevented by normalizing each feature to vary in the same numerical range. Mathematically feature normalization is expressed as

$$X_{ik}^{normalized} = \frac{X_{ik} - \mu_k}{\sigma_k}, \tag{4.1}$$

where μ_k and σ_k are the mean and standard deviation for feature $k \in \{1, ..., 42\}$ over all feature vectors i. The mean and standard deviation values used to normalize the training data are stored to be used later in the segmentation phase. This is important because the features are needed to stay consistent during training and testing phases and across multiple images.

4.3.2 Training

This section will study two logistic regression classifiers along the regularization path: A classifier that has the highest performance and a more sparse classifier that has a higher degree of sparsity but slightly worse performance.

If a lot of data is available, one can simply divide it into three data sets, namely *training data*, *validation data* and *test data*. First various classifiers corresponding to different λ values can be trained using the training set. Then the performance of

each classifier is assessed by testing them against the validation set. Based on the obtained results, a classifier having a desired λ value and performance is selected. Finally, the test set is used to measure the classification performance of the chosen classifier.

However, as in this case, the amount of available data is scarce so the selection of λ is done using *cross-validation* (CV). CV is an iterative method that allows to reduce the size of validation set without producing noisy estimates of the desired λ value [20]. Now only two data sets are needed: Training set and test set. In the beginning of CV, the training data is divided into non-overlapping subsets. During one iteration, also known as *fold*, one of these subsets is used for validation and the rest for training after which classification performance is evaluated using some performance measure. The iterations are continued as long as each subset has been used once for validation. The results from the iterations are recorded and averaged on which the selection of λ is based.

Implementation of the training phase is conducted in two steps: Cross-validation and final training. In the cross-validation step, *leave-one-image-out* cross-validation is performed, where the feature data from one image is considered a subset. During each iteration, a logistic regression classifier is trained in MATLAB using `lassoglm` function. In fact, the whole regularization path is computed where a distinct classifier is trained for each provided lambda value. A set of 100 predefined lambda values were chosen to give enough resolution for selecting the desired degree of sparsity. Each of these classifiers are then validated against the fold's test image and AUC as a function of lambda is recorded. Once the iterations are finished, the AUC measurements are averaged so that a graph is obtained where an average of 10 AUC values corresponds to one λ value. In the final training step, regularization path is trained using all the 10 training images resulting in 100 classifiers each having different degree of sparsity depending on the lambda value. Now two of these classifiers are selected for the segmentation phase.

Selection is based on average of AUC computed during cross-validation. The lambda value that corresponds to the highest AUC value in the average graph is selected as the *maximum AUC classifier* $\boldsymbol{\beta}_{max}$. If there are multiple such classifiers, the most sparse one is selected. The second classifier is the *sparse classifier* $\boldsymbol{\beta}_{sparse}$, which is chosen according to one-standard-error rule [22]. The rule specifies that the mean of AUC measurements for the classifier may be at most one standard deviation lower than that of the maximum AUC classifier. The standard deviation is calculated from the AUC measurements of the maximum AUC classifier over the 10 CV iterations.

The obtained classifiers have the following properties:

- The maximum AUC classifier: 31/42 nonzero coefficients, $\|\boldsymbol{\beta}_{max}\|_1 = 11.8029$

- The sparse classifier: 6/42 nonzero coefficients, $\left\|\boldsymbol{\beta}_{sparse}\right\|_1 = 1.4527$.

Both of these classifiers along with the regularization path from the final training are shown in Figure 4.3. The classifiers have been visualized by dashed red lines in the figure.

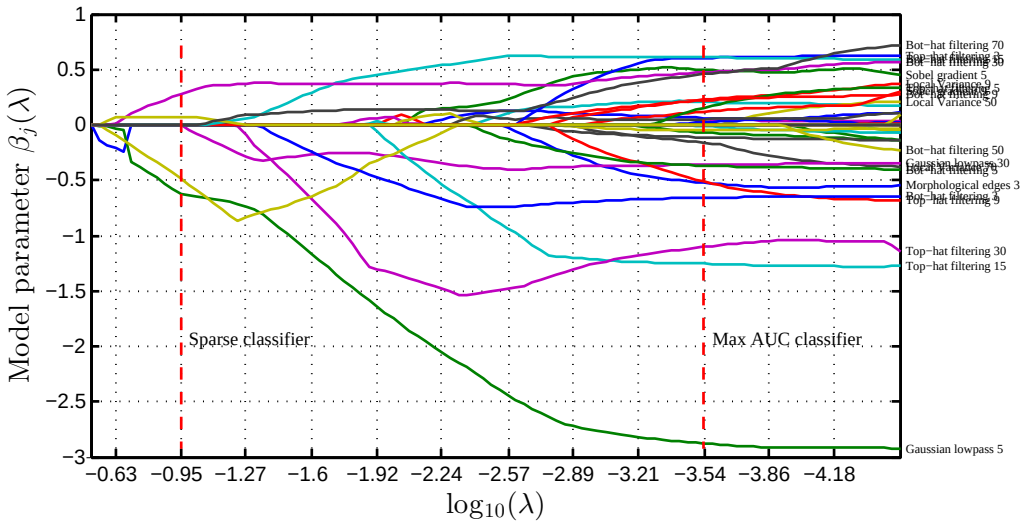

Figure 4.3: The regularization path of the final training. Every λ corresponds to a classifier. One graph corresponds to a feature and visualizes its importance in prediction at varying regularization levels. Feature names are shown in the right side of the figure. The left side of the figure shows coefficient values when the amount of regularization is high and only small number of features are allowed to participate in prediction. In the right side, the coefficient vector $\boldsymbol{\beta}$ is less constrained and more features are allowed to participate. The two selected classifiers are presented in dashed vertical red lines.

Each graph corresponds to one feature and the name of the feature has been printed on the right side of the figure. This type of presentation allows easy interpretation of the trained classifiers. Features that have high absolute coefficient value $\beta_j(\lambda)$ are the most important in classification because the coefficient is a multiplier for the actual feature value. In practice, features that have high positive coefficient value are used directly in the classification whereas high negative coefficients invert the feature values after which they are added to produce the linear predictor output.

Suppose training of a classifier where the regularization parameter λ is set to a high value and gradually decreased while more and more features are allowed in prediction. The left side of the figure shows the features that are picked first for the prediction task. These are Gaussian low-pass filters of window sizes 3,5 and 9. Window size 3 has the highest coefficient magnitude and is seen as the blue graph that first declines from the horizontal axis. If only one feature was allowed, this would be the best among the feature pool for logistic regression. This is because microporosity is represented in higher frequencies than big porosity and is discarded

during the filtering. The resulting blurry image is a good approximator of the pore locations since pores have low intensities and everything else has high intensity values. This is inverse of what is desired which is why the feature coefficient is negative.

Decreasing λ slightly, includes top-hat filtering and bot-hat filtering of disk radius 50 in the prediction. Top-hat filter is the yellow line that crosses the sparse classifier at -0.5 and bot-hat filter is the other yellow line having positive coefficient at the same λ position. Top-hat filtering of disk radius 50 is large enough to detect the walls (gives high values to walls) between the pores whereas bot-hat filtering of same size directly detects pores (gives high values to pores). For this reason, bot-hat filters have positive and top-hat filters negative coefficients.

To better understand why certain features are chosen in the automatic selection process, sample images of features are shown in Figure 4.4. Above every image patch is the name of the feature along with the coefficient value used in the maximum AUC classifier. The figure displays the 20 most influential features that are ordered according to descending coefficient magnitude. The same coefficient values are shown also in the previous regularization path image. The image patches in the bottom row are: original image, linear combination of the features in the figure, sigmoid transform of the linear combination, *i.e.*, the probability image, thresholded probability image and the ground truth.

The highest coefficient magnitude is given to Gaussian low-pass filtering having window size of 5. This is reasonable as it discards microporosity. The second largest in magnitude is Top-hat filtering having disk radius of 15. Top-hat filtering detects small hills in the image. In this case, it detects the granularity of the walls giving high values when such grains are present and low values when not, *e.g.*, at pore locations. For this reason, it has a negative coefficient in order to gain high values at pore locations. The same reasoning applies also for the next top-hat filter. In general, features that have negative coefficient detect undesired phenomena whereas positively valued features detect desired phenomena.

The linear combination of the presented feature images is depicted in the bottom row. It shows that summing and multiplying all the feature images gives high values at the pore locations and resembles the ground truth image after sigmoid transform.

4.3.3 Segmentation Procedure

This section describes the method to segment new, previously unseen SEM images of zeolite. The class label of each pixel in a new image is predicted using one of the previously trained classifiers. A block diagram illustrating the steps is depicted in the bottom part of Figure 4.2.

The segmentation phase begins with the same exact preprocessing procedure that

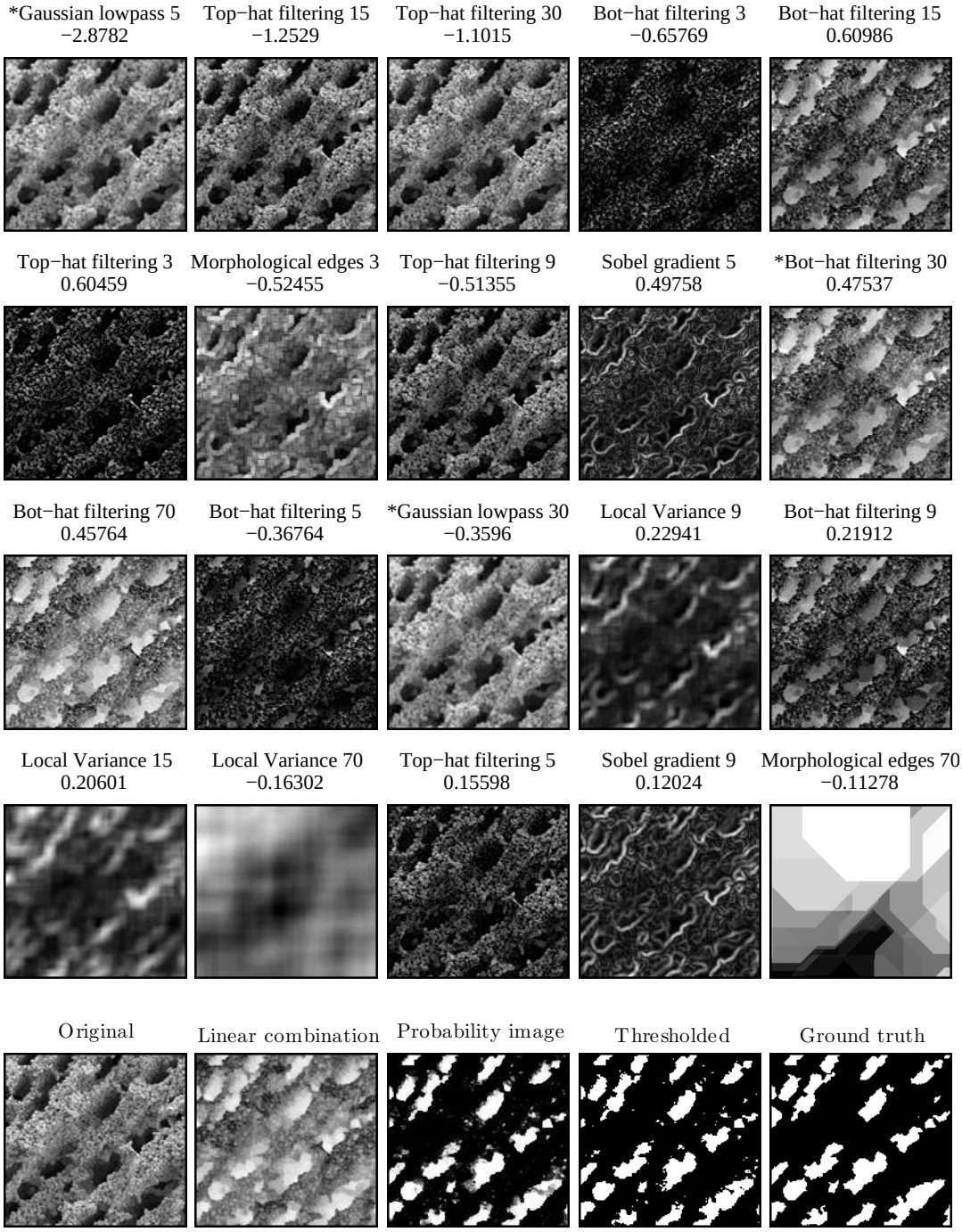

Figure 4.4: Image patches visualizing the 20 most influential features in the maximum AUC classifier. Above a patch is the name of the feature and its coefficient value. The features have been ordered according to descending absolute coefficient value, *i.e.*, the highest coefficient is in the top left corner. The bottom row contains images: Original, linear combination of the features in the image, sigmoid transformed linear combination (probability image), thresholded probability image and the ground truth. Notice that not every feature is shown here but the 20 most influential. Features hat have an asterisk * before the name are also present in the sparse classifier.

was used in the training phase in order to have the data in the same form as in training phase. This includes CLAHE processing, feature extraction and normalization providing feature matrix $\mathbf{X}$.

After preprocessing, probabilities for new pixels are obtained using equation 3.18. In practice, dot product between parameter vector $\boldsymbol{\beta}$ and a row from feature matrix $\mathbf{X}$ is computed and added to the intercept term β_0 giving the linear predictor output for one pixel. Applying sigmoid function to the output of the linear predictor gives the class label probability of the corresponding pixel. Predicting every pixel in the same manner yields the probability image.

Examples of probability images are shown in Figure 4.5 where maximum AUC prediction image and sparse prediction image are probability images given by the maximum AUC classifier and the sparse classifier respectively. Bright areas indicate

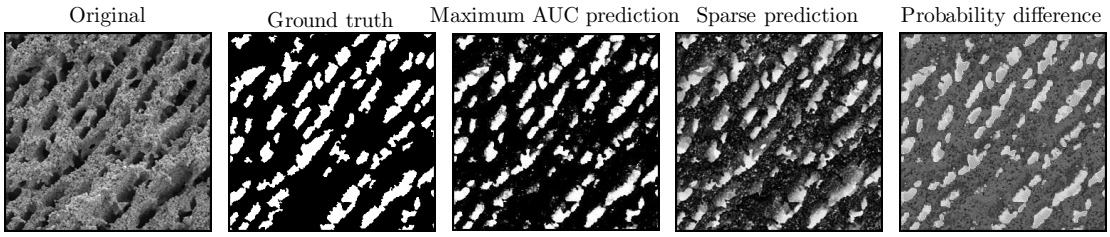

Figure 4.5: Comparison of the probability images given by the maximum AUC classifier and the sparse classifier. The rightmost image is the difference between the two predictions.

positions where the probability of a pore is high whereas dark regions are more likely to be regions of material. This probability image is thresholded by 0.5 because any higher value means that a pore is more probable than the background. This operation provides a binary image which indicates the locations of pores.

The main difference between the two classifiers is certainty of predictions, which is seen in the difference image. The probabilities given by the maximum AUC classifier are much higher than those of the sparse classifier. Moreover, the sparse classifier is less certain about microporosity and yields higher probabilities for it more easily which is seen as noise like speckles and more grayish background in comparison with the maximum AUC prediction.

4.4 Results

The performance of the designed method was evaluated from two different aspects, namely classification performance and segmentation performance. The first considers the classifiers ability to predict pore locations separately from the rest of the framework, and the second analyzes the system's segmentation ability as a whole.

Classification Results

The two trained classifiers were tested against a distinct test set of 8 images. Classification results were collected and ROC graphs based on all classifications were constructed for both classifiers. These ROC graphs are presented in Figure 4.6 along with the respective AUC values. As the figure shows, the problem is described well

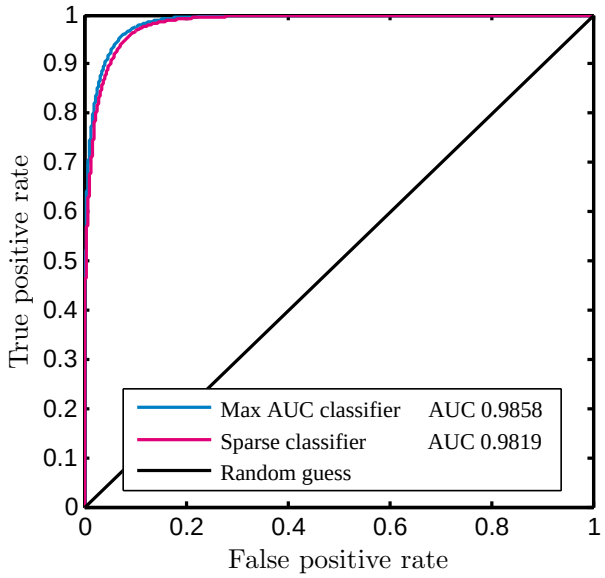

Figure 4.6: ROC graphs of the two trained classifiers. Also the AUC values are shown in the legend.

with only six features and the performance of the sparse classifier is close to that of the maximum AUC classifier. This indicates that the classification problem is fairly uncomplicated with the given feature set since almost as good solution is found using only a small subset of the features.

Additionally, the total number of correctly classified samples, *i.e.*, *accuracies* of both classifiers were calculated to give more intuitive presentation of the classification performance. For accuracy computation, the probability images are thresholded by 0.5 and the acquired results are presented in Table 4.2.

Some amount of error is present in the results due to subjectivity of the annotation process. Sometimes human annotator might consider a phenomenon in the image as a pore and sometimes not. This leads to inconsistencies in the training and test data which could impair classification ability of a classifier. However, such inconsistencies become less meaningful in reality if the classifier generalizes well and is not learning peculiarities of the training data.

Table 4.2: Classification accuracies of the maximum AUC classifier and the sparse classifier

Image	Max AUC classifier	Sparse classifier
Image 1	0.968	0.915
Image 2	0.968	0.914
Image 3	0.960	0.925
Image 4	0.955	0.903
Image 5	0.939	0.859
Image 6	0.902	0.939
Image 7	0.933	0.875
Image 8	0.901	0.875
Average	**0.941**	**0.901**

Segmentation Results

Two empirical discrepancy methods were chosen to evaluate the segmentation performance. In such method, the segmentation result is compared against a human segmented ground truth image. It is important to notice that even though a comparison between ground truth and segmentation result is carried out without human interaction, empirical discrepancy method is still subjective since a ground truth image is the result of a subjective evaluation by one or several annotators. [40]

Pore segmentations in this case were evaluated individually by separating each segment from the background and comparing it to the corresponding ground truth object. It would have been possible to compare the results with the ground truth image by simply comparing two binary images containing all the segments, but in order to stay consistent with case 2, individual approach was chosen to have a consistent and simple evaluation procedure.

The quality of the segmentation results was evaluated using two metrics which give no importance to true negative classifications. This is crucial because the number of true negative instances dominates the binary image in which a pore has been extracted from the background. The first metric is PAS metric [41] that measures how well two regions overlap. In this case, how well a ground truth object overlaps with a segment given by the algorithm. It is written as

$$PAS = \frac{TP}{TP + FP + FN} \tag{4.2}$$

where TP denotes the number of true positives, FP false positives and FN false negative pixels. In effect, this means dividing the overlapping area of both regions

by the total area covered by both regions. In other words, it is the area of the intersection divided by the area of the union.

The other metric is F_1-score which has the form

$$F_1 = \frac{2TP}{2TP + FP + FN}. \tag{4.3}$$

From the definitions it is seen that F_1-score is very similar to PAS metric but gives true positives twice the weight.

Comparison between segmentation result and the ground truth was done by iterating through every object in the ground truth image and comparing it to every segment in the segmentation result. The segment which gave the highest PAS score was chosen as the corresponding particle in the segmentation results. The obtained scores using the maximum AUC classifier are shown in Table 4.3. Some ground truth

Table 4.3: Case 1 segmentation results. Every row contains the obtained results from one test image. PAS and F_1 columns show the average scores of comparing each segment in an image to the corresponding ground truth segment.

Image number	PAS average	F_1-score average	Not found	Found	True number
Image 1	0.690	0.814	21	492	338
Image 2	0.713	0.827	32	665	667
Image 3	0.803	0.896	10	506	433
Image 4	0.707	0.845	63	2122	1618
Image 5	0.837	0.910	2	219	135
Image 6	0.839	0.921	1	324	185
Image 7	0.805	0.894	5	373	107
Image 8	0.796	0.894	3	674	265
Average	**0.774**	**0.875**			

objects did not have a corresponding region in the segmentation results. These objects were counted and provided in "Not found" column of the table. In addition, the total number of objects found by the method is presented in "Found" column and "True number" is for the number of objects in the ground truth image.

The results show that the algorithm finds more objects than human annotators are able since the difference between found and true number is high in comparison with not found objects. A visual examination of the segmentation results confirmed this as the pores found by the algorithm are plausible choices for pores.

An example of a segmentation result is displayed in Figure 4.7 where the orig-

inal image patch is in the top left corner. Top middle and top right images are

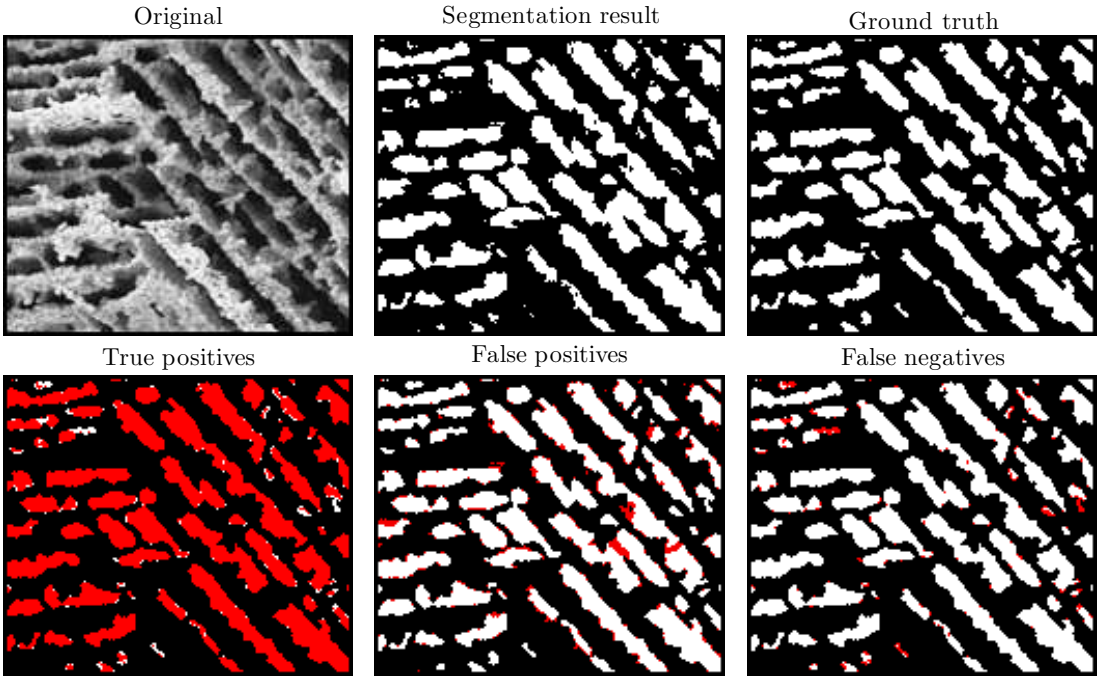

Figure 4.7: Segmentation result comparison with ground truth. From left to right and top to bottom: Original, segmentation result, ground truth, true positive segmentations, false positives in red and false negatives in red color. The white areas in the bottom row images denote ground truth.

the segmentation result and ground truth images respectively. Bottom row images visualize true positive, false positive and false negative pixels superimposed in red color on the ground truth image.

Many false segmentations are due to subjectivity of drawing boundaries for pores which are seen as red contours on the white ground truth objects. In these cases, human annotator marks the pores more or less differently every time causing inconsistency in the ground truth segmentations whereas the method is able to segment pores in a consistent manner. The problem is present mostly in locations where a smooth transition of intensities make it difficult to determine one annotation that is better than all the other choices.

As stated, no absolute truth exists in segmentation of zeolite pores due to subjective nature of defining a pore. For this reason, the numbers presented in Table 4.3 may be rather seen as a general indicator of the segmentation performance than the absolute truth. In fact, the numbers represent the ability of the method to arrive at one subjective predefined goal among many other possible goals that are equally correct as this one. Therefore, it is reasonable to say that the method works better than the numeric results indicate.

The average time to segment one image with the maximum AUC classifier is 3.8 seconds and 1.7 seconds with the sparse classifier. The computation times were measured in Windows 8.1 running on Intel i7-3517U processor.

5. CASE 2: PARTICLE SEGMENTATION OF TEM IMAGES

Traditional electron microscopy analysis of nanomaterials include many stages from sample preparation to manually measuring the desired characteristics from produced microscope images. One example of such analysis task is measuring the average particle size in a nanosilver sample.

Manual calculation of average particle size is a repetitive and tedious operation which can be aided by automatic image analysis. Automation allows acquisition of more consistent information of materials at a faster pace than manual segmentation. Sometimes, even previously unavailable information is acquired, *e.g.*, in situations where the work has been left undone due to excessive amount of work needed to complete it. Therefore, this chapter considers segmentation of nanosilver particles of electron microscope images for which an automatic image segmentation method is developed, implemented and tested.

5.1 Data

The images in this case were taken with Transmission electron microscope (TEM) which is another common imaging method in the field of materials science. In Transmission electron microscopy, an electron beam of uniform current density is emitted through a very thin specimen. The interaction between the specimen and the beam alters the current density of the beam according to the properties of the specimen. The altered beam is then captured and stored as an image. [42]

In total there are 9 TEM images of silver nanoparticles which were provided by the Department of Materials Science of Tampere University of Technology. All the images have dimensions 2688×2672.

Annotation of the images was done using open source image editing program GIMP 2.8.14 [43] in which every particle was manually extracted from the background. In effect, this process is manual segmentation resulting in a set of binary images that hold the label information of the particles. The annotated images were divided into two distinct sets: Training data and test data. The training data comprise 7 images and the test set 2 images.

5.2 Objective

Segmentation of silver nanoparticles has much in common with segmentation of zeolite pores. The aim is to automatically extract each particle in a TEM image from the background and store the location information into a binary image. The main difference with the previous case is that the particles may overlap in the images which makes segmentation task more challenging.

An example TEM image of nanosilver is presented in the left image of Figure 5.1 where each dark and round object in the image is a silver nanoparticle. The right

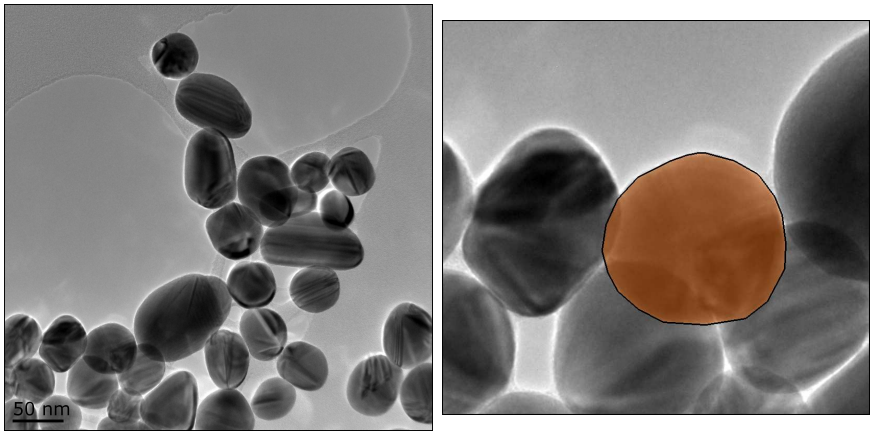

Figure 5.1: A transmission electron microscope image of silver nanoparticles is in the left image. The right image shows an ideal segmentation of one particle.

image shows the general idea of how each particle should be segmented where one particle has been separated from the rest of the image in orange color. Overlapping sections of the neighboring particles can be seen through the transparent color. The task is challenging as overlapping particles eliminate the use of the simplest segmentation methods such as various types of intensity thresholding. Additionally, some edges of the particles are not visible for human viewer which makes automatic segmentation even more challenging.

5.3 Implementation

The developed methods in case 1 and 2 are similar in many aspects. To avoid repetition, the focus is on the differences of these methods and only a brief recapitulation is given of the similar steps.

The method developed for particle segmentation is also training based. Unlike in the previous case, a classifier is trained to discriminate between edge pixels and non-edge pixels rather than directly predicting the class labels. The edge probability

image given by logistic regression is then used as an input for seeded watershed segmentation. Again, the choice of classifier is logistic regression for it is fast and produces models that are easy to interpret.

The block diagram of the method is depicted in Figure 5.2 where the upper part shows the steps of the training phase and the lower part shows the steps of the segmentation phase. The regions of lighter color correspond to preprocessing and

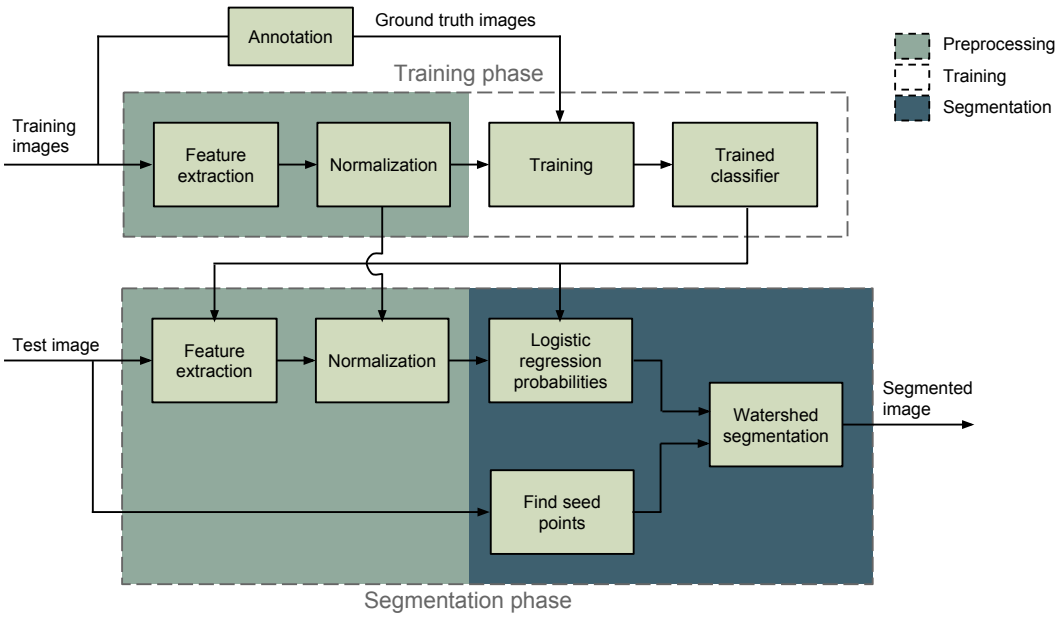

Figure 5.2: A block diagram of the case 2 segmentation pipeline. The areas on the lighter green background denote preprocessing and the darker color denotes the actual segmentation phase.

the regions of darker color denotes the actual segmentation. The partitioning of the segmentation phase has the same structure as shown in Figure 2.2.

The arrows between training and segmentation phases are exactly the same as in case 1. The arrow from the trained classifier block to the feature extraction block denotes classifier's sparsity information and the arrow between two normalization blocks denotes the use of same normalization parameters in both phases.

5.3.1 Preprocessing

The first step in preprocessing is resizing in which image dimensions are reduced by 50%. The dimensions are reduced in order to keep the amount of data reasonable without having significant effect on the meaningful information. Here, the important information is the boundaries of the particles which should be sharp and well distinguishable after the reduction.

The second step is feature extraction. In total 44 features, which is two more than

in the previous case, are extracted and shown in Table 5.1. Since CLAHE is not used

Table 5.1: The complete pool of features used in the training phase. From these features, the best are chosen automatically during training. Unlike in case 1, CLAHE is now used as a feature.

Feature	Parameter	Values
Local Variance	window size	3,5,9,15,30,50,70
Sobel gradient	kernel size	3,5,9,15,30,50,70
Morphological edges	disk radius	3,5,9,15,30,50,70
Gaussian low-pass	window size	3,5,9,15,30,50,70
Top-hat filtering	disk radius	3,5,9,15,30,50,70
Bot-hat filtering	disk radius	3,5,9,15,30,50,70
CLAHE	number of blocks	5×5, 15×15

in the preprocessing step, it is included to the pool of features to improve generality of the feature set. CLAHE is not needed in the preprocessing phase because the TEM images have fairly even illumination and the amount of contrast variation is minimal. As a feature, it functions the same way as the others. The input image is processed with CLAHE resulting in an image in which each pixel contains the feature value of the corresponding original pixel. The parameters were chosen to be 0.01 for the clipping level and the window size presented in the table is the number of blocks each image is divided into.

Now all the features are extracted by image domain filtering. The resulting images are vectorized and concatenated horizontally yielding the feature matrix for one image. This is repeated for all the images and the obtained feature matrices are concatenated vertically giving the big feature matrix $\mathbf{X}$ of size $15\ 064\ 896 \times 44$. It is good to notice that the amount of data is approximately 5 times higher than in case 1.

The final step in preprocessing phase is feature normalization is identical to the one performed in case 1 (4.3.1). This operation normalizes the features to have zero mean and unit standard deviation.

5.3.2 Training

Training a classifier for silver nanoparticle edge prediction has the same goal as previously obtaining two classifiers with different degree of sparsity. The first one is a classifier that has the best performance and the second one is a less complex sparse classifier having slightly worse performance.

At first, classifiers were trained while the class imbalance was considerably high. In fact, the average percentage of edge pixels in the training images is 5.39% which has the effect of ML to find a solution that predicts every sample to the non-edge class. Regarding this, it has been studied that training with imbalanced classes has similar effect on classification as setting imbalance to the classification error costs [44; 45]. For example, reducing samples from one class reduces also its misclassification error cost. The found solution implies that the misclassification error cost for edge class is so low that the best solution is to ignore the class altogether.

This behavior can be avoided by balancing the amount of samples in the classes, *i.e*, *under-sampling*. As the amount of edge samples is scarce, the number of non-edge samples is reduced for training. The manner in which the samples are selected is based on randomization. For each image in the training set, the number of edge samples are counted and an equal amount of non-edge samples are randomly picked from the image and the rest are discarded. This operation results in 1:1 edges-to-non-edges ratio which solves the initial problem, and a better ML estimate is found. Under-sampling the non-edge class reduces the amount of training data giving feature matrix $\mathbf{X}$ of dimensions $1\,267\,127 \times 44$.

However, using this classifier in practice can be problematic as the found ML solution assumes both classes to be equally probable. As a consequence, thresholding the probabilities of new samples by 0.5 will result in many false positive classifications as the operating conditions of the classifier are different. This might not be a big issue in practice since the segmentation method does not threshold the probability image but uses it directly, and the highest probabilities should be induced by the true edge pixels. ROC graph together with AUC value is a useful performance measure in this case since it does not operate on a single threshold level but rather all of them. This is why it is the choice of performance measure for cross-validation.

Training of the classifiers is conducted in two steps: Cross-validation and final training. Both of these steps were computed in MATLAB using the *Glmnet* package [26] due to fast algorithm [46]. This package would have been used in the previous case as well, but some bugs prevented its efficient use and `lassoglm` was selected instead.

The CV step is otherwise the same as previously, but now the number of subsets in the training data is 7 where one subset corresponds to one image. During each iteration, AUC of every classifier is recorded as a function of λ. Once the iterations are finished, the measurements are averaged so that a graph is obtained where an average of 7 AUC measurements corresponds to one λ value.

In the final training step, a regularization path is trained using all 7 images. As previously, two classifiers are selected: The maximum AUC classifier $\boldsymbol{\beta}_{max}$ and the sparse classifier $\boldsymbol{\beta}_{sparse}$. The first one is selected based on the maximum value in the

averaged AUC graph, and the second is chosen according to the one-standard-error rule.

The obtained classifiers have the following properties:

- The maximum AUC classifier: 41/44 nonzero coefficients, $\|\boldsymbol{\beta}_{max}\|_1 = 18.1606$
- The sparse classifier: 16/44 nonzero coefficients, $\|\boldsymbol{\beta}_{sparse}\|_1 = 1.9039$

These classifiers have been plotted together with the regularization path from final training in Figure 5.3. where each coefficient has been visualized as a function of

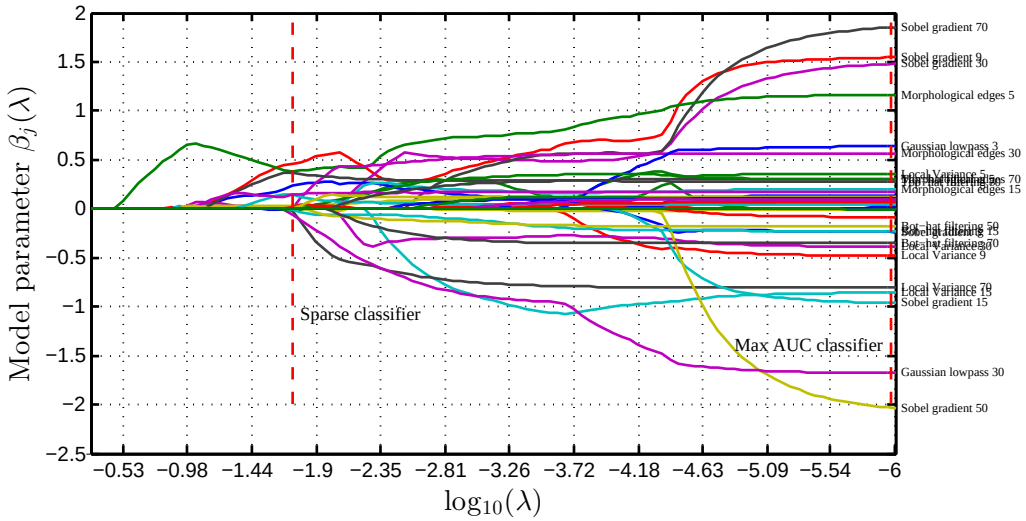

Figure 5.3: The regularization path of final training. The classifiers are denoted by dashed red lines. The graphs visualize coefficient values of features at various levels of regularization. Feature names are shown in the right side of the figure.

lambda and the name of the feature is printed on the right side of the figure. The left side of the figure shows model coefficients when the amount of regularization is high, *i.e.*, the L_1-norm of the model coefficient vector is small.

Suppose training of a classifier where the λ parameter is set at a high value and begins to decrease gradually. The first feature allowed in prediction is morphological edge filter with disk radius of 5. It is the green graph that rises first from the horizontal axis. As λ decreases more, both morphological edge filters and top-hat filters of small radii begin to get nonzero coefficients. At the λ value of the sparse classifier, local variance with window size of 9 has become the feature that has the highest influence in the prediction. It is the topmost red graph at the position of sparse classifier.

A visualization of the most important features is depicted in Figure 5.4, where each feature has been presented along with its name and coefficient value. The images are sorted according to coefficient magnitude so that the upper left feature has

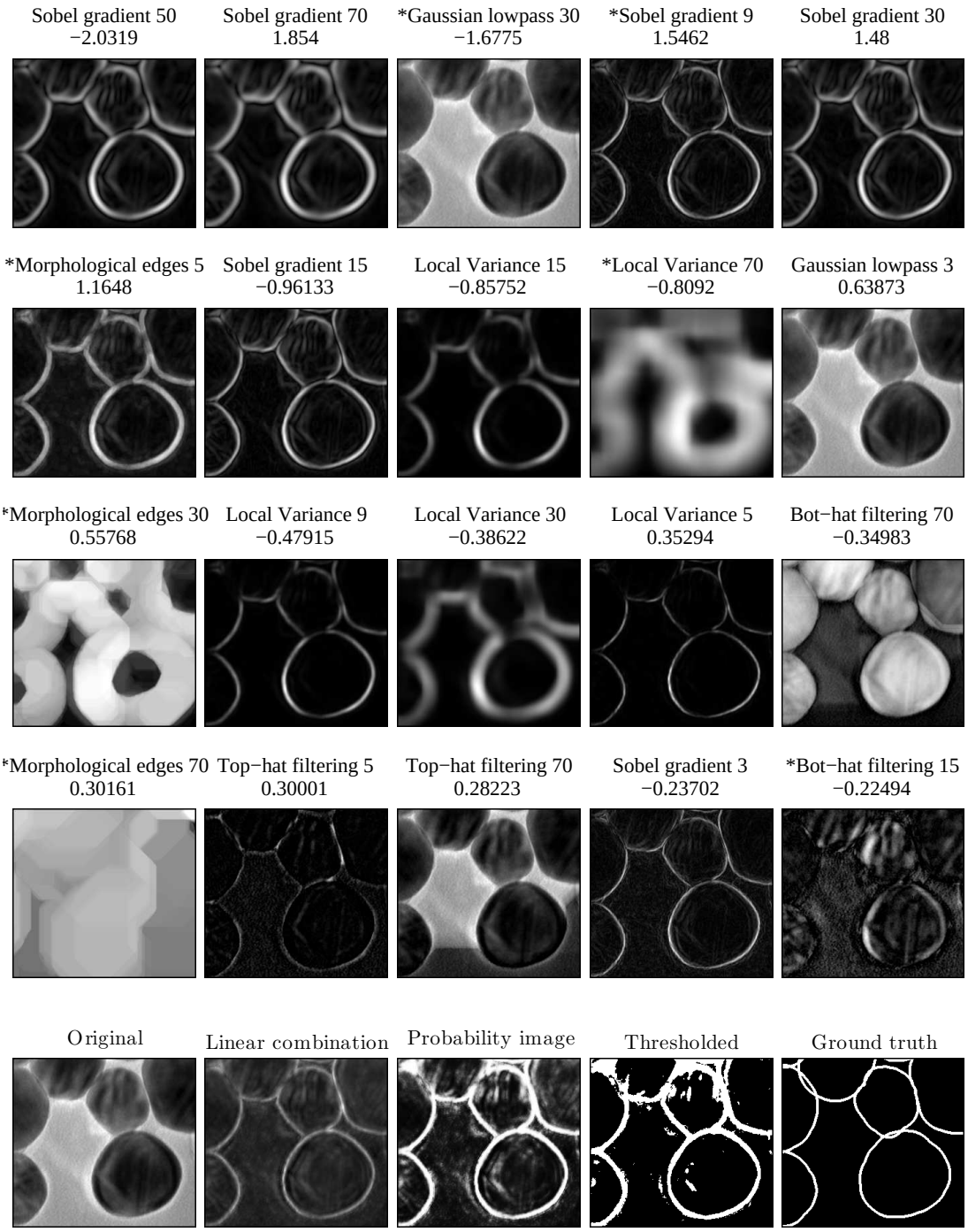

Figure 5.4: Image patches visualizing the 20 most influential features in the maximum AUC classifier. The name of the feature and its coefficient value is presented above the patches. Ordering of the features is based on descending absolute coefficient value, *i.e.*, the highest coefficient is in the top left corner. The bottom row contains images: Original, linear combination of the features in the image, sigmoid transformed linear combination (probability image), thresholded probability image and the ground truth. Features hat have an asterisk * before the name are also present in the sparse classifier.

the highest magnitude. The coefficients in the image are the same as in maximum AUC classifier, which can be seen as well from the previous regularization path image 5.3. The image patches in the bottom row are: Original image, linear combination of the coefficients and features, probability image (sigmoid transformed linear combination), thresholded probability image using threshold 0.5 and the ground truth.

Perhaps counterintuitively, the first feature is a Sobel edge detector of window size 50 having a negative coefficient value. Closer examination reveals that linear combination of all the positively valued edge detection features generate blurry edges which become sharper as negatively valued edge detectors are added.

The third feature is a Gaussian low-pass filter with window size of 30. Since the coefficient is negative, adding this feature to the linear predictor output increases the probabilities of edges when the gray level is low, *i.e.*, at particle locations. The reason being that it is more common for the annotations to be on top of a dark particle than the background due to overlapping particles. This phenomenon is learned by the classifier giving higher edge probability at the position of a dark gray level. The effect is also seen from the linear combination image where particle interiors are generally lighter.

In the bottom row of the figure, sigmoid transforming the linear combination gives the probability image which is similar to the ground truth. The expected false positive classifications, due to under-sampling, are seen as noise near the particle edges as well as in the thickness of the edges.

5.3.3 Segmentation Procedure

The most prominent differences between case 1 and case 2 lie in the segmentation phase that is also seen from the block diagram in Figure 5.2. The general idea of the method is to segment previously unseen TEM images of nanosilver by applying seeded watershed segmentation. The watershed method is performed on an edge probability image given by logistic regression classifier and the seed points are searched using a combination of distance transform, morphological filtering and watershed transform.

Edge Probability Image

Edge probabilities in a TEM image are predicted using the previously trained classifiers. Before edges may be predicted, the new image needs to be preprocessed. Preprocessing is done in the same manner as in the training phase in section 5.3.1 where features are extracted and normalized yielding feature matrix $\mathbf{X}_{new}$. Every feature needs not to be extracted because some coefficients in the parameter vector

$\boldsymbol{\beta}$ are zero, which reduces the computational demand.

Once features have been extracted, the probabilities of edges are predicted by calculating dot product between each row in $\mathbf{X}_{new}$ and the model parameters $\boldsymbol{\beta}$. The product is then added to the intercept term β_0 and sigmoid transformed to determine the probability of an edge for the corresponding pixel. The same operation is performed for each row in the feature matrix and the resulting probabilities are reordered to form an image.

Examples of the probability images are presented in Figure 5.5 where the edges in the leftmost image are predicted using the two previously trained classifiers. The second image from the left is an edge probability image given by the maximum AUC classifier and the third one is predicted using the sparse classifier. The rightmost image is a difference image obtained by subtracting the third image from the second one. The difference image reveals that the maximum AUC prediction tends to give

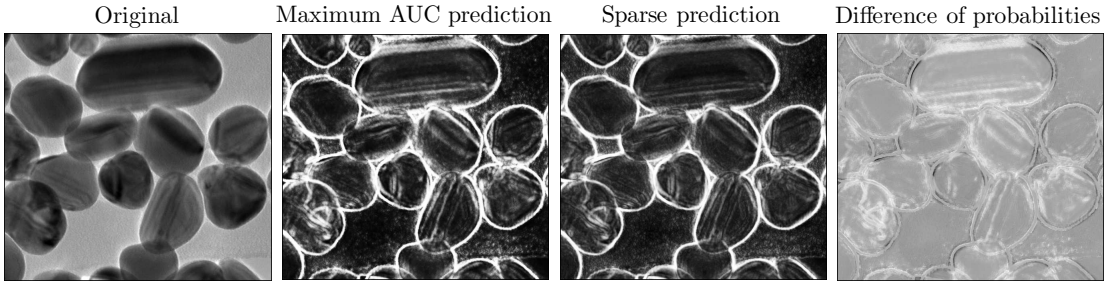

Figure 5.5: A comparison between the probability images obtained by the two classifiers. The images are: Original, probability image given by the maximum AUC classifier, probability image given by the sparse classifier and the difference between the two probability images.

higher probabilities for the particle textures which are seen as the white regions. The classifier also exhibits more precise and narrow edges which is seen as the dark rims around strong edges. Perhaps the most important difference is the certainty of predictions on overlapping particles. Having higher edge probabilities lowers the risk of segmenting two particles under a single segment. For this reason, the maximum AUC classifier is selected as the classifier for the following steps since it predicts overlapping edges more accurately.

Experiments showed that low-pass filtering the probability image yields smooth edges in the produced segments. The operation made the filtered edge correspond better with the actual border of the particle. However, this was not confirmed by any quantitative measure. The choice of filter is Gaussian low-pass filter $\sigma = 2.5$ and window size 20×20. This operation provides the final edge probability image that is given as an input to the watershed algorithm.

Finding Seed Points for Particles

The general idea behind seed point searching is to find one seed point for each particle in the image because watershed segmentation creates exactly as many segments as there are seed points. Since every pixel must belong to either the watershed lines or one of the catchment basins, a separate seed region is generated for the background as well to prevent including background in the particle segments.

Seed point searching for particles begins with thresholding the original image using Otsu's method [27]. It is a histogram based segmentation method that finds a threshold level that minimizes the within class variance of the two created histogram partitions. It has been shown that this is equivalent to maximizing the between class variance. Thresholding produces an initial segmentation S where foreground particles (denoted by 1) are extracted from the background (denoted by 0). An example image of Otsu's thresholding method is shown in the middle of Figure 3.15, where the original image patch is in the left.

Particles in the TEM images are often connected to each other, which has the effect of forming connected regions in the initial segmentation. Some of these connections can be removed by morphological opening which also reduces noise from the binary image induced by global thresholding. The structuring element was chosen not to remove the smallest particles in the image, therefore a disk with radius of 10 was found suitable for the task. The binary image resulting from this operation is denoted by S_c.

Next, the inverse of S_c is distance transformed. The output is a grayscale image D where the background regions have value 0 and object regions have varying gray levels depending on the distance to the nearest background pixel. The transform produces highest values near the centers of the objects and can be effectively used to determine points within objects even if the object border is not entirely visible.

The seed points for watershed segmentation are simply chosen to be local maxima in the distance transform D. For finding the local maxima, grayscale morphological dilation proves useful. The operation to find maxima is expressed mathematically as

$$M = D - (D \oplus B), \tag{5.1}$$

where B is the structuring element. Here the dilated distance transform image is subtracted from itself, which sets local maxima to value 0. This is because grayscale dilation does not alter their values as it is effectively maximum filtering. The resulting image M may be then thresholded to obtain a binary image M_b containing the seed points. A good initial value for the size of the structuring element is approximately the average particle size even though overlapping might cause situations where two pinnacles of the distance transform fit within the structuring element.

This can be controlled by reducing the size of the structuring element. In this case, a disk shaped structuring element of size 50 was used.

It is important to notice that choosing a large structuring element for searching maxima does not restrict multiple maxima to be found within the structuring element since the distance transform itself may have the same maximum value at various locations within a small area. This is the case for many objects in the images.

Because distance transform calculates the distance to the nearest white pixel, all background pixels get value zero. As an undesired consequence, these positions are regarded as local maxima as well if they form a large enough continuous area that fits the structuring element. A visualization of the undesired local maxima in M_b is seen in the left image of Figure 5.6 as the large red regions. A straightforward

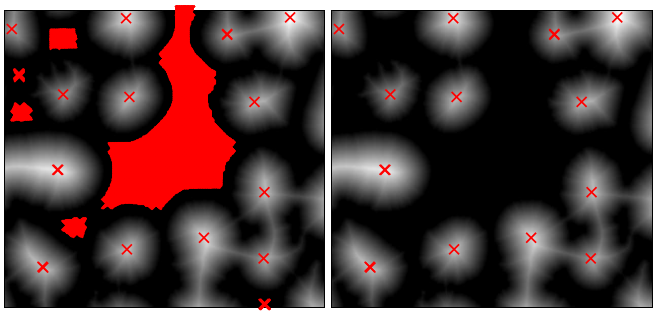

Figure 5.6: Local maxima in a distance transform denoted by red crosses. The left image is the result M_b after morphological local maxima search. The right image shows the filtered result which is based on initial segmentation.

approach to remove these regions is to allow only maxima that belong to foreground based on the initial segmentation S. This operation yields the final seed image and it is depicted in the right image.

Finding Seed Points for the Background

Another seed image is constructed for the background in order to have one continuous region for the background after watershed segmentation. For this, a *Voronoi diagram* is computed. Voronoi diagram is a partitioning of an image according to a set of points. A partition contains all the pixels of the image that are closer to one point than any other point. An important property of the diagram is that the divide lines are connected if the points are not lying on a straight line. [47]

Meyer showed that watershed transform of a distance image is equal to the Voronoi diagram [48]. Therefore, the Voronoi diagram of the initial segmentation S_c is obtained by distance transforming the inverse of it and applying watershed transform. Instead of using single points in the partitioning, each distinct object in

the initial segmentation is paired with a region. The divide lines of the watershed transform are the Voronoi diagram for S_c.

An example of the Voronoi diagram of an initial segmentation is presented in Figure 5.7 in red color. The diagram is chosen as the seed region for the background.

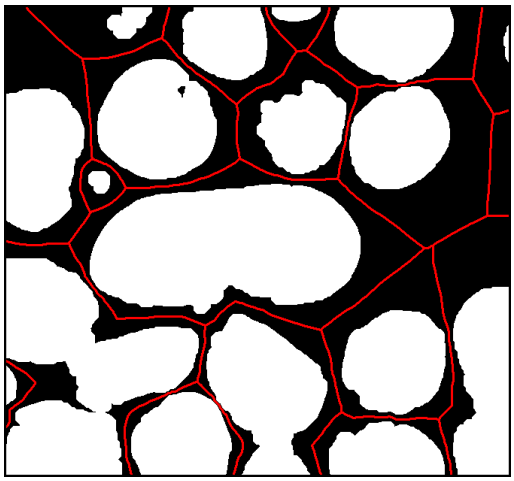

Figure 5.7: Background seed region obtained by computing Voronoi diagram of an initial segmentation. The divide lines are shown in red color.

Now that the seed points for background and particles are selected, they are given to the watershed method, which separates nanoparticles from the background.

Watershed Segmentation

The actual segmentation of particles is performed using seeded watershed segmentation on a blurred edge probability image. The two seed images obtained in the previous sections are combined together by logical OR operation to conveniently hold all the needed points in one image.

In MATLAB, seeded watershed segmentation is achieved by modifying the input image rather than giving the seed points as parameters for the `watershed` function. To do this, MATLAB provides the function `imimposemin`, which modifies the input image to have local minima only in specified locations. It takes two images as parameters: The image to be modified and a binary image indicating the locations the seed points. The algorithm is based on morphological reconstruction, which is not considered in this work.

An example of an image modified by `imimposemin` is presented in Figure 5.8, where the left image shows the found seed points superimposed on the blurred edge probability image. The locations of the seed points are now the only local minima in the right image, and the traditional unseeded watershed segmentation is executed

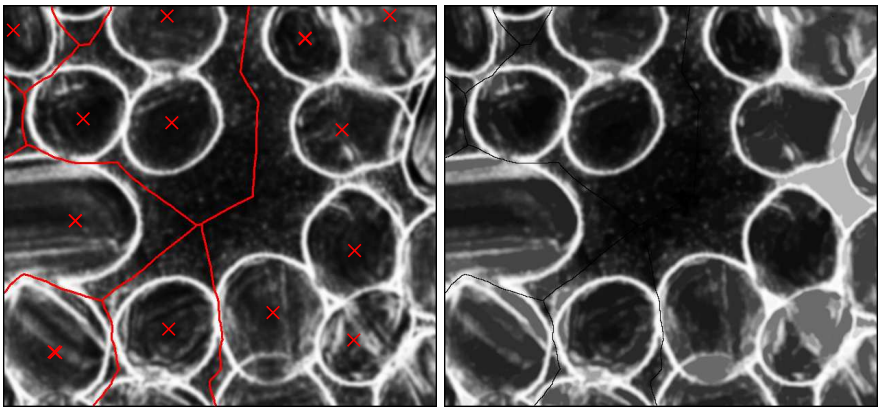

Figure 5.8: The left image shows the continuous background seed points (red line) and particle seed points (crosses) superimposed on the blurred edge probability image. For visualization purposes the background seed points have been dilated. The right image is the modified probability image where the only local minima are in the positions of particle seed points and background seed region.

on it. This is effectively seeded watershed segmentation and it partitions the image into particle and background regions according to the specified local minima.

5.4 Results

This chapter is divided into two sections where the performance of the developed method is evaluated from two different aspects. In the first section, classification performance is evaluated independently from the rest of the framework, and the second section considers segmentation ability of the whole procedure.

Classification Results

Both of the trained classifiers were tested against the test set of 2 images. The classification results were collected from the two images for both classifiers, and ROC graphs were constructed based on the acquired results. These are shown in Figure 5.9, where the corresponding AUC values are also provided. The figures suggest that the classification problem, with the given feature set, is more challenging than in case 1. This is seen from the relative decrease in the AUC score between the maximum AUC classifier and the sparse classifier. The reason behind is that efficient discrimination of new samples simply requires more information which indicates that the problem is more complex.

The accuracies of the classifiers were not considered in this case because true negative samples dominate the class prior probabilities. Consequently, classifying every sample to non-edge class, *i.e.*, misclassifying every edge pixel would yield accuracy of approximately 94%. This percentage is also known as the chance level.

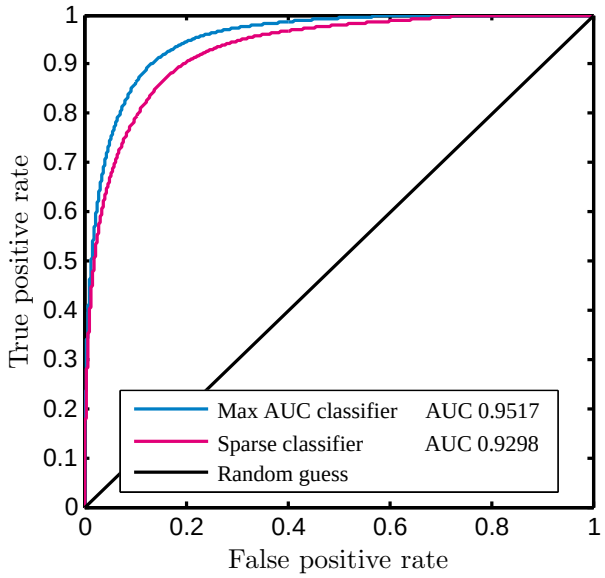

Figure 5.9: ROC graphs of the two trained classifiers together with AUC values.

Segmentation Results

The method in assessing segmentation performance is otherwise the same as in previous case, but here some preparation is needed. The preparation step removes two types of segments from the segmentations to gain more informative results.

The first type of segments originate from partitioning of the background. Even though the method aims to create one uniform segment for the background, sometimes more than one is produced. This occurs when separate objects in the initial segmentation are located approximately on a line, and are not surrounded by other particles, such as the three topmost particles in Figure 5.1. In a case like this, the Voronoi diagram produces a partition that spans across the image. For such partition, the divide line is cut by the image borders resulting in a non-continuous background seed region. This leads to creation of two distinct regions for the background as watershed creates a separate segment for each disconnected seed region. Therefore, to avoid ground truth objects from having background as the corresponding segmentation, all the background regions were manually selected and removed. This is because measuring the overlap between a background segment and a ground truth object is not meaningful in this work.

The second type of removed segments are regions of area less than 50 pixels. These segments are due to the Meyer flooding algorithm, used by MATLAB's `watershed` function. The algorithm has tendency to create tiny undesired segments [37]. In reality, such tiny regions are not plausible choices for a nanoparticle segments and are thus not considered in the results.

After the preparation, the segmentation performance is evaluated in the exact

same way as in the previous case, where the ground truth objects are compared to the corresponding region in the segmentations. The correspondence is determined by the highest PAS score between a ground truth object and a segment. The obtained PAS scores and F_1 scores are presented in Table 5.2, where each row holds the average score from one test image. The bottom row shows the average of all the test images. The table contains also counts of objects that have no corresponding

Table 5.2: Case 2 segmentation results. Every row contains the obtained results from one test image. PAS and F_1 columns show the average scores of comparing each segment in an image to the corresponding ground truth segment.

Image number	PAS average	F_1-score average	Not found	Found	True number
Image 1	0.698	0.832	3	68	81
Image 2	0.820	0.896	6	24	32
Average	**0.759**	**0.864**			

particle in the segmentations. This is shown in the "Not found" column. The "Found" column displays the number of distinct segments and the "True number" is the actual number of particles in the images.

Particles are considered not found when the PAS score is below 0.05 because some particles overlap with segments that successfully segment another particle in the image. In this case it is obvious that the particle is not found even though it is overlapping with a segment. A common attribute for these situations is a small nonzero PAS score and simple thresholding solves the problem.

The sum of the "Not found" and "Found" columns is not equal to "True number" because some particles have the same corresponding segment. Another cause for the difference between the two numbers is oversegmentation of particles. These two errors are among the five most noticeable errors present in the segmentations. Examples of these are shown in Figure 5.10, where the erroneous segments are superimposed in red color on the original TEM image.

The leftmost image in the figure is an example of two nanoparticles sharing a segment. This occurs when local maxima are searched from the distance transform and only one maxima is found for the two overlapping particles.

The second image from the left shows a segment that has some additional background included. This error occurs because overlap of the particles has the effect of forming enclosed background areas in the initial segmentation. This prevents the background seed region from reaching these enclosed areas, thus including the enclosed area to some of the surrounding particle segments.

The third image is an error due to pronounced texture patterns in the particle. Sometimes strong textures split objects during initial segmentation, which leads to

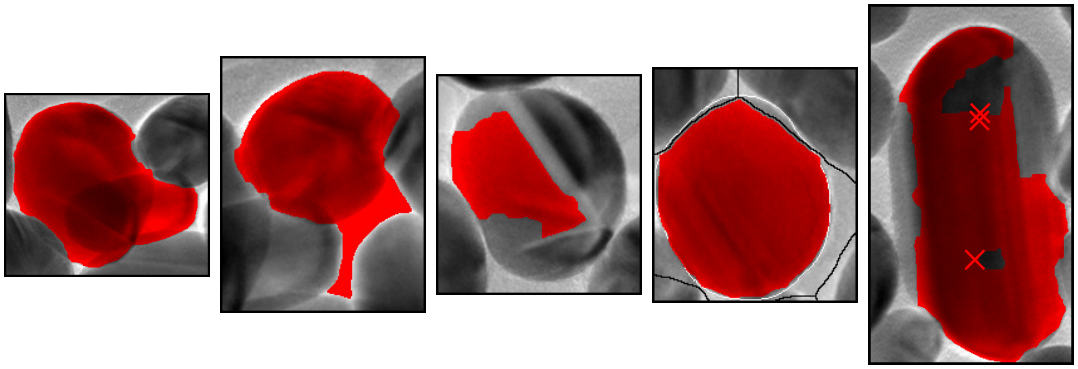

Figure 5.10: Examples of the five most common segmentation errors.

finding two seed points within one particle. Some other time, the textures resulted in high edge probabilities causing partial segmentation during watershed segmentation.

The fourth image is an error where the background seed region overlaps with a particle. In the image, true borders of the particle are denoted by a white line and the background seed is denoted by a black line. In this case, the particle region is not able to expand over the background seed which is why the segmentation completes erroneously.

The fifth type of error originates from rod shaped particles. If the shape of the particle is elliptic, more than one local maximum is found within the particle. The rightmost image shows the three seed points within an elliptic particle.

An example segmentation is presented in Figure 5.11 , where a faint line is drawn in the image to assess the quality of segmentations in two different scenarios. The upper part of the test image shows segmentation performance when the particles are not overlapping. Within this part, the average PAS score of the segmentation is 0.83815. The bottom part of the image contain particles having much more overlap where the average PAS score is 0.56392. Together with visual examination, it is evident that the method performs well if the particles have low amount of overlap and not as well when the amount of overlap is high.

It is observed that the segments created in high overlap conditions tend to be larger than the average particle size. Conversely, in the case of little overlap, the produced segments are slightly smaller than the corresponding ground truth objects. This is explained by the fact that error types of 1 and 2 are more common in clustered regions whereas types 3, 4 and 5 are more common in non-clustered regions. However, it is good to notice that these results are derived from a small amount of test data, and to gain more definitive results, more tests should be performed.

As opposed to case 1, the annotation process is less subjective and the errors caused by it are rather small. This is because the images are of good quality, and there are not many options on how to draw the boundaries for a particle. On

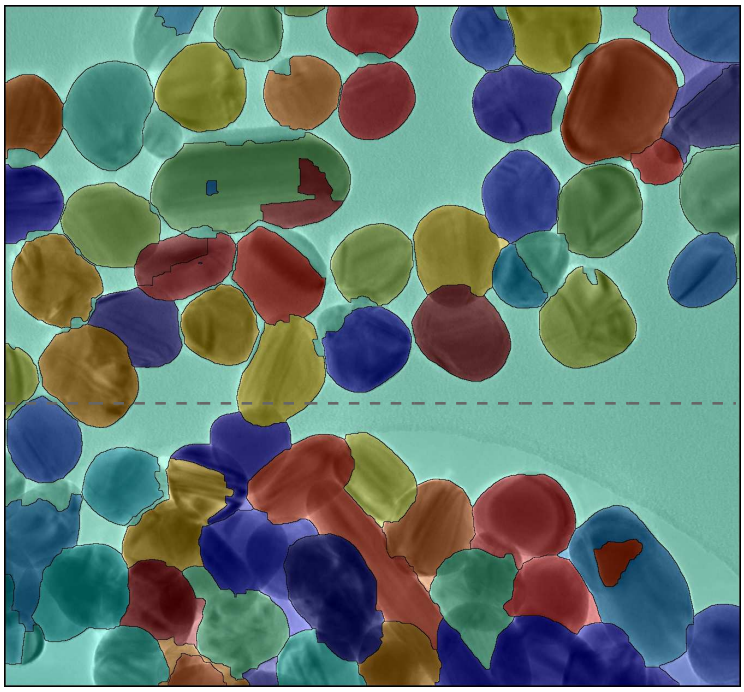

Figure 5.11: Segmentation result for test image 1. For visualization purposes, each segment has been given a random color. The dashed line is drawn to divide the image into two partitions based on the amount of overlap. PAS score for the upper segmentation is 0.83815 and for the lower part 0.56392

the other hand, in clustered regions, particle boundaries are hidden behind other particles which required some guessing to be done during the annotation process. Moreover, it was sometimes difficult to determine which borders belong to which particle in a cluster and some mistakes might have been made.

On average, segmentation of one image took 23.8 seconds when using the maximum AUC classifier and 14.2 seconds with the sparse classifier. The computation times were measured in Windows 8.1 running on Intel i7-3517U processor. It is clear that the method, using either of the classifiers, is much faster than manual segmentation.

6. CONCLUSION

This thesis is a case study of two image segmentation problems in the field of materials science. In both cases, automatic segmentation methods were developed and tested. Based on the obtained results, the methods can successfully segment electron microscope images of zeolite pores and silver nanoparticles.

The method from case 1 exhibits solid performance and is able to find more pores faster and with better consistency than humans. A comparison between the produced segments and human annotated ground truth segments yields average PAS score of 0.79 and F_1 score of 0.89. Much of the error is due to ambiguity of the annotation process which is why it is reasonable to assume that the method performs even better than what the results imply.

The segmentation results given by the case 2 method are satisfactory for particles which have small amount of overlap. However, in heavily clustered regions, the segmentations are flawed. The measured average scores are 0.76 for PAS metric and 0.86 for F_1 score. A notable drawback of the method is its inability to properly segment overlapping particles, though, by avoiding clustered particle samples, the method is able to produce fine segments much faster than manual segmentation.

Some improvement ideas were considered for the second case during the final stages of the work. It could be beneficial to study further the amount of undersampling of the training data to have it correspond better to the true operating conditions of the classifier. It is likely that it would have positive effect on the probabilities within particles that have strong texture patterns. Another idea to decrease probabilities within particles is to experiment with sampling methods other than randomly selecting the negative samples. As an example, selecting more difficult samples from particle interiors where there are strong texture patterns could reduce the edge probabilities in similar cases. Furthermore, the linearity of the classifier is probably causing some errors in the classification phase. Some nonlinearity could be introduced to the system by augmenting features. In fact, a simple experiment revealed that the classification performance is improved 2 percentage points by augmenting a subset of the features to the second power.

Generally speaking, the task of image segmentation is ambiguous if it is not known beforehand what needs to be extracted. For instance, silver nanoparticles may be segmented in two different ways: Extracting each particle individually or all of them

as a group. This is the reason why information of the desired outcome needs to be somehow inserted into the segmentation method. Conventionally, the information is introduced already in the selection of the algorithm, in the parameter choices and general assumptions of the problem which requires expertise in the field of image analysis. Training based segmentation methods are different in the sense that anyone who knows what is to be extracted, can collect the training data containing information of the segmentation target. In other words, the designing phase is not solely in the hands of the expert but the person who gathers the data as well.

What makes it interesting is combining it with sparsity promoting regularization. As explained in the work, sparsity promotion allows the use of general purpose pool of features from which the best features are drawn according to the problem. In simpler terms, this means that the method is able to adapt to the segmentation task without requiring any additional user interaction, thus making its operation suitable for anyone regardless of background.

This evoked an idea of creating a semi-automatic image segmentation software. The software would allow users to collect training data from images of interest by clicking or painting over the chosen pixels. Choosing LASSO regularized logistic regression classifier as the core functionality of the software would have two major benefits. The first one is feature selectivity making the classifier more adaptive and suitable for various tasks. The second one is the probabilistic output of logistic regression. In order to produce class labels, simple thresholding is applied where the threshold level can be controlled by the user. And because the threshold level is not known in advance, the validation process could be based on AUC, which evaluates classifiers over all threshold levels. Similar software already exist, such as Ilastik [49], where the feature selection is left for the end user. This however, requires knowledge of the task as well as the features to be used efficiently.

The algorithms developed during this thesis had a central role in calculating pore parameters of zeolite structures in the referred publication [50]. The work put into this thesis produced also an interactive segmentation software *TUT Nanoparticle Annotator* for the Department of Materials Science of Tampere University of Technology.

REFERENCES

[1] Williams, D. B., Carter, C. B. Transmission Electron Microscopy: A Textbook for Materials Science. 2nd ed., 2009, Springer Science & Business Media. 760 p.

[2] Ojuva, A., Akhtar, F., Tomsia, A. P., Bergström, L. Laminated adsorbents with very rapid CO2 uptake by freeze-casting of zeolites. ACS Applied Materials & Interfaces 5(2013)7, pp. 2669–2676.

[3] Damm, C., Münstedt, H., Rösch, A. The antimicrobial efficacy of polyamide 6/silver-nano-and microcomposites. Materials Chemistry and Physics 108(2008) 1, pp. 61–66.

[4] Roduner, E. Size matters: Why nanomaterials are different. Chemical Society Reviews 35(2006)7, pp. 583–592.

[5] Ruusuvuori, P., Manninen, T., Huttunen, H. Image segmentation using sparse logistic regression with spatial prior. Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European. 2012, pp. 2253–2257.

[6] Liu, T., Jurrus, E., Seyedhosseini, M., Ellisman, M., Tasdizen, T. Watershed merge tree classification for electron microscopy image segmentation. Pattern Recognition (ICPR), 2012 21st International Conference on. 2012, pp. 133–137.

[7] Wong, H., Head, M., Buenfeld, N. Pore segmentation of cement-based materials from backscattered electron images. Cement and Concrete Research 36 (2006)6, pp. 1083–1090.

[8] Cheng, H.-D., Jiang, X., Sun, Y., Wang, J. Color image segmentation: Advances and prospects. Pattern Recognition 34(2001)12, pp. 2259–2281.

[9] Megason, S. G., Fraser, S. E. Imaging in systems biology. Cell 130(2007)5, pp. 784–795.

[10] Kotsiantis, S., Kanellopoulos, D., Pintelas, P. Data preprocessing for supervised leaning. International Journal of Computer Science 1(2006)2, pp. 111–117.

[11] Duda, R. O., Hart, P. E., Stork, D. G. Pattern Classification. 2nd ed., 2012, John Wiley & Sons. 680 p.

[12] Meijering, E. Cell segmentation: 50 years down the road. Signal Processing Magazine, IEEE 29(2012)5, pp. 140–145.

[13] Laudate, T. Imaging guidelines for scanning electron microscopy: With a few basic guidelines, a novice SEM operator can learn to produce high-quality images. Advanced Materials and Processes 161(2003)pp. 23–25.

[14] Gonzalez, R. C., Woods, R. E. Digital Image Processing. 3rd ed., Upper Saddle River, NJ, USA 2006, Prentice-Hall, Inc. 954 p.

[15] Pizer, S. M., Amburn, E. P., Austin, J. D., Cromartie, R., Geselowitz, A., Greer, T., Haar Romeny, B., Zimmerman, J. B., Zuiderveld, K. Adaptive histogram equalization and its variations. Computer Vision, Graphics, and Image Processing 39(1987)3, pp. 355–368.

[16] Haralick, R. M., Sternberg, S. R., Zhuang, X. Image analysis using mathematical morphology. Pattern Analysis and Machine Intelligence, IEEE Transactions on (1987)4, pp. 532–550.

[17] Shih, F. Y. Image Processing and Mathematical Morphology: Fundamentals and Applications. 2009, CRC press. 440 p.

[18] Vincent, L. Morphological grayscale reconstruction in image analysis: applications and efficient algorithms. Image Processing, IEEE Transactions on 2 (1993)2, pp. 176–201.

[19] Dougherty, E. R., Lotufo, R. A. Hands-on Morphological Image Processing. vol. 71, 2003, SPIE press. 272 p.

[20] Bishop, C. M. Pattern Recognition and Machine Learning. 1st ed., vol. 4, 2006, Springer New York. 740 p.

[21] Fawcett, T. An introduction to ROC analysis. Pattern Recognition Letters 27 (2006)8, pp. 861–874.

[22] Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Friedman, J., Tibshirani, R. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed., vol. 2, 2009, Springer. 745 p.

[23] Michie, D., Spiegelhalter, D. J., Taylor, C. C. Machine learning, neural and statistical classification. Ellis Horwood Series in Artificial Intelligence 1(1994) .

[24] Tibshirani, R. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological) (1996)pp. 267–288.

[25] Hebiri, M., Lederer, J. How correlations influence lasso prediction. Information Theory, IEEE Transactions on 59(2013)3, pp. 1846–1854.

[26] Qian, J., Hastie, T., Friedman, J., Tibshirani, R., Simon, N. Glmnet for MATLAB [WWW]. [Accessed on 16.03.2015].
Available at: `http://web.stanford.edu/~hastie/glmnet_matlab/`.

[27] Otsu, N. A threshold selection method from gray-level histograms. Automatica 11(1975)pp. 285–296.

[28] Ramer, U. An iterative procedure for the polygonal approximation of plane curves. Computer Graphics and Image Processing 1(1972)3, pp. 244–256.

[29] Wang, S., Kubota, T., Siskind, J. M., Wang, J. Salient closed boundary extraction with ratio contour. Pattern Analysis and Machine Intelligence, IEEE Transactions on 27(2005)4, pp. 546–561.

[30] Horowitz, S. L., Pavlidis, T. Picture segmentation by a directed split-and-merge procedure. Proceedings of the Second International Joint Conference on Pattern Recognition. 1974, pp. 433.

[31] Brice, C. R., Fennema, C. L. Scene analysis using regions. Artificial Intelligence 1(1970)3, pp. 205–226.

[32] Haralick, R., Kelly, G. Pattern recognition with measurement space and spatial clustering for multiple images. Proceedings of the IEEE 57(1969)4, pp. 654–665.

[33] Chuang, K.-S., Tzeng, H.-L., Chen, S., Wu, J., Chen, T.-J. Fuzzy c-means clustering with spatial information for image segmentation. Computerized Medical Imaging and Graphics 30(2006)1, pp. 9–15.

[34] Xu, C., Prince, J. L. Snakes, shapes, and gradient vector flow. Image Processing, IEEE Transactions on 7(1998)3, pp. 359–369.

[35] Shi, J., Malik, J. Normalized cuts and image segmentation. Pattern Analysis and Machine Intelligence, IEEE Transactions on 22(2000)8, pp. 888–905.

[36] Beucher, S., Meyer, F. The morphological approach to segmentation: The watershed transformation. Mathematical Morphology in Image Processing. 1993, pp. 433–481.

[37] Roerdink, J. B., Meijster, A. The watershed transform: Definitions, algorithms and parallelization strategies. Fundamenta Informaticae 41(2000)1, pp. 187–228.

[38] Meyer, F., Beucher, S. Morphological segmentation. Journal of Visual Communication and Image Representation 1(1990)1, pp. 21–46.

[39] Goldstein, J., Newbury, D., Joy, D., Lyman, C., Echlin, P., Lifshin, E., Sawyer, L., Michael, J. Scanning Electron Microscopy and X-ray Microanalysis. 3rd ed., 2003, Plenum Publishers. 690 p.

[40] Zhang, Y. J. A survey on evaluation methods for image segmentation. Pattern Recognition 29(1996)8, pp. 1335–1346.

[41] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., Zisserman, A. The pascal visual object classes (voc) challenge. International Journal of Computer Vision 88(2010)2, pp. 303–338.

[42] Reimer, L., Kohl, H. Transmission Electron Microscopy: Physics of Image Formation. 5th ed., vol. 36, 2008, Springer Science & Business Media. 590 p.

[43] GNU Image Manipulation Program (GIMP) [WWW]. [Accessed on 16.03.2015]. Available at: `http://www.gimp.org/`.

[44] Maloof, M. A. Learning when data sets are imbalanced and when costs are unequal and unknown. ICML-2003 Workshop on Learning From Imbalanced Data Sets II. 2003, pp. 2–1.

[45] Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J. Classification and Regression Trees. 1984, CRC press. 368 p.

[46] Friedman, J., Hastie, T., Tibshirani, R. Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software 33(2010) 1, pp. 1–22.

[47] Aurenhammer, F. Voronoi diagrams - a survey of a fundamental geometric data structure. ACM Computing Surveys (CSUR) 23(1991)3, pp. 345–405.

[48] Meyer, F. Topographic distance and watershed lines. Signal Processing 38 (1994)1, pp. 113–125.

[49] Sommer, C., Straehle, C., Kothe, U., Hamprecht, F. A. ilastik: Interactive learning and segmentation toolkit. Biomedical Imaging: From Nano to Macro, 2011 IEEE International Symposium on. 2011, pp. 230–233.

[50] Ojuva, A., Järveläinen, M., Bauer, M., Keskinen, L., Valkonen, M., Akhtar, F., Levanen, E., Bergstrom, L. The mechanical performance and CO2 uptake of ion-exchanged zeolite A structured by freeze-casting. Journal of the European Ceramic Society (2014).