University of West Bohemia
Faculty of Applied Sciences
Department of Cybernetics

MASTER THESIS

Pilsen, 2013

Bc. Jakub Prüher

University of West Bohemia
Faculty of Applied Sciences
Department of Cybernetics

Master Thesis

Tracking of Changes in Indentation of Polymers from Image Data

Pilsen, 2013

Bc. Jakub Prüher

Declaration

I hereby declare that this master thesis is completely my own work and that I used only the cited sources.

. . . .

Pilsen, May 16, 2013 Bc. Jakub Prüher

Acknowledgements

I would like to thank my thesis supervisor, Ing. Marek Hrúz, for his guidance. I am indebted to my parents for providing necessary material and spiritual support during my studies. I am grateful to my brother for backing me up financially when buying my new PC. Finally, my thanks go to Joan DeForeest, M.A. for her willingness to take on the task of correcting my English.

Abstract

The process of non-destructive polymer quality assessment involves an inspection under a microscope by a skilled operator. To make the quality assessment process automatic, a series of images capturing the polymer indentation recovery must be obtained and analyzed. This thesis deals with the problem of tracking of changes in polymers from image data. The image segmentation process is identified as an integral part of such a problem. Support vector classification is employed to segment the images. Features based on variance are proposed and their performance is evaluated. The designed features proved to be suitable for the segmentation of indentation images as evidenced by the high classification performance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the world of today, as technological development progresses, increasing demands on the manufacturing process and quality assurance are only to be expected in many areas of industry. One of the important areas is polymer quality inspection. Polymers find their usage in all manner of applications ranging from spacecraft manufacture to the production of chewing gum. The basic building block of biological life, DNA, is also a polymer (biopolymer).

Currently, testing of polymer quality usually involves destruction of the manufactured part and thus can be quite costly. In the area of metal hardness testing, there is an array of non-destructive hardness measuring techniques. Metal hardness is tightly related to other properties of the materials [2], [7]. What if these non-destructive techniques could be utilized to measure the properties of polymers? In [26], an indentation-based technique is used to quantify the viscoelastic response of the bulk polymers.

Several material hardness measurement methods exist. All of these are based on an indenter of varying shape being forced into the surface of the tested specimen with a predefined load. From the shape of the indenter, the created impression and load applied, hardness is calculated. Depending on the shape of the indenter, these methods are named after their inventors. These include Rockwell, Brinell, Knoop and Vickers hardness tests. For microindentation measurements, Knoop and Vickers methods are used.

This thesis deals with the problem of automatic computer vision-based analysis of Vickers indentations in polymers. In automatic inspection of indentations, the aim is to extract dimensions of the impression from its microscopic image. Some automatic inspection methods try to locate the center of the indentation [14], while others aim to find the boundaries of the indentation [13], effectively segmenting out the indentation from the background. Hrúz, Široký and Maňas [17], utilize particle swarm optimization to find the boundaries of the Vickers impression. Other Vickers indentation measurement techniques utilize wavelet theory [31] and the Hough transform [30].

A wide variety of image segmentation approaches and techniques is available today. The simplest of which is thresholding. Techniques based on splitting and merging of the parts of the image based on predefined criteria are collectively referred to as split-and-merge methods. Other available approaches to segmentation include methods based on graph-cuts, clustering and classification. For the purposes of this thesis, an approach based on Support Vector Machine (SVM) classification was chosen.

Organization of this thesis is as follows. Metal hardness testing according to Vickers is discussed in Chapter 2. Chapter 3 outlines some of the fundamental image segmentation approaches, including classification. The SVM classifier and its historical development is described in Chapter 4. The practical part of this thesis starts with Chapter 5, where the process of feature extraction is described. In subsequent chapters, the choice of the method of testing is justified and results are presented and discussed. The final chapter provides the conclusion for this thesis.

Chapter 2

Vickers Hardness Test

One of the important properties of materials that helps to identify them is hardness. Askeland [2] mentions that "Hardness can not be defined precisely. Hardness, depending upon the context, represents resistance to scratching or indentation and a qualitative measure of the strength of the material." In order to determine the hardness of metals, several methods of hardness testing have been developed. Among others, these include: Rockwell, Brinell, Knoop and Vickers hardness tests. In this chapter, only the Vickers hardness testing method is described.

2.1 Origins

In 1921 in the UK at Vickers Ltd, Smith and Sandland developed the Vickers test as an alternative to the Brinell test for measuring hardness of materials. As Vander Voort and Fowler [27] explain, "The test was developed because the Brinell test (introduced in 1900), which (until recently) used a round hardened steel ball indenter, could not test steels harder than $\sim$ 450HB ($\sim$ 48 HRC)." Historically, the Vickers hardness measurement was separated into two categories: microhardness and macrohardness measurements. When the applied load on the indenter is less than 1000 gf, microhardness is measured. In contrast, when the load is more than 1 kgf, macrohardness is measured. Because in macro tests the size of the indentation is larger, "... macro tests yield a gross product average, while micro tests indicate variations in hardness" [27]. Micro Vickers tests are generally performed in a laboratory, whereas macro tests are performed in the workshop.

2.2 The Method

Like most other hardness measurements methods, the Vickers hardness test is based on forcing an indenter of predefined shape into the surface of the measured specimen. This results in an impression being created in the material, which is subsequently inspected by a skilled operator with the help of a magnifying lens. The relevant dimensions of the impression are measured and used to calculate the hardness number.

The indentation-based hardness measurement methods can be distinguished from one another by the shape of the indenter used. In Vickers hardness test, a regular, pyramid-shaped indenter with an angle of 136° between opposing faces and the length of the diagonal d is used. As the indenter must be harder than the material measured, it is usually made of diamond. The Surface area of the resulting indentation A in square millimeters can be

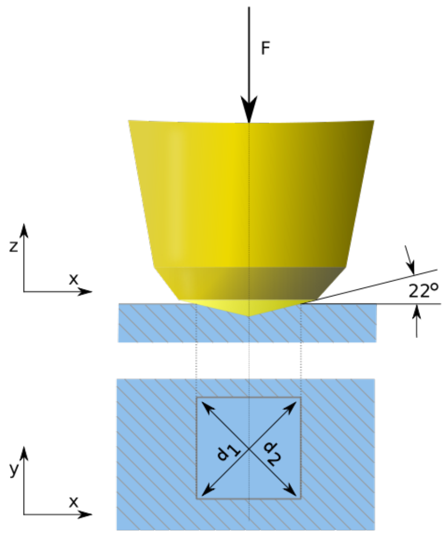


Figure 2.1: Vickers hardness measurement scheme.

determined as

$$A = \frac{d^2}{2 \cdot \sin(136°/2)} \approx \frac{d^2}{1.8544}, \tag{2.1}$$

where d is the average diagonal length of the indentation in the material. The corresponding Vickers hardness HV is given by the ratio

$$HV = \frac{F}{A} = \frac{1.854F}{d^2}, \tag{2.2}$$

where the force F is in kgf (kilogram-force is the force magnitude exerted by one kilogram in the Earth's gravitational field), and diagonal length d is in millimeters. As the diagonal lengths of the impression can be different from one another (which can be caused by the indenter being offset from the normal direction to the surface), the operator measures the length of both diagonals of the impression and uses the average value in hardness number calculation.

Unlike in other hardness tests, in the Vickers test the load is applied smoothly and held in place for the duration of $10 - 15$ seconds. The hardness number is given in the format xHVy (e.g. 440HV30) or xHVy/z (e.g. 440HV30/20), if the time during which the load is applied differs from the standard $10 - 15$ seconds. In the above hardness number format, x is the hardness number, y is the load applied in kgf and z is the load time.

Unlike the Rockwell test, which uses a number of different scales in different situations, an advantage of the Vickers test is that it uses only one scale for all materials. On the other hand, Rockwell tests are fully automated and the process of hardness determination does not require the participation of a skilled operator.

Chapter 3

Image Segmentation

Image segmentation is one of the key problems in computer vision. It concerns itself with partitioning an image into a number of segments, where pixels in each segment share some visual characteristics, such as brightness, texture, color or motion. The goal of segmentation is twofold. The first goal is to decompose the image into meaningful parts that usually correspond to real-world objects. The second goal is to provide simplified image representation for further high-level processing. It is thus a building block for all subsequent processes like shape analysis and object recognition [23]. In [15], the state of the theory of image segmentation is described as follows: "There is no theory of image segmentation. Image segmentation techniques are basically ad-hoc and differ precisely in the way they emphasize one or more of the desired properties and in the way they balance and compromise one desired property against another."

Various approaches and techniques for image segmentation have been proposed over the years; however, in this chapter only a few basic techniques are discussed. For a comprehensive overview of image segmentation techniques, the reader is referred to [24]. For earlier methods, [3] can be recommended and for more recent approaches, [25, 24] can be consulted.

3.1 Thresholding

Image thresholding is the earliest and the simplest method of image segmentation. “If background lighting is arranged so as to be fairly uniform, and we are looking for rather flat objects that can be silhouetted against a contrasting background, segmentation can be achieved simply by thresholding the image at a particular intensity level” [11].

Formally, let $I(i,j)$ be a 2D array of brightness values representing the grayscale image. The goal of thresholding is to find an optimal threshold value T, for the purposes of generating a segmented binary image $I_T(i,j)$ such that

$$I_T(i,j) = \begin{cases} 1 & \text{if } I(i,j) \geq T, \\ 0 & \text{if } I(i,j) < T. \end{cases} \tag{3.1}$$

Value 1 is assigned to those pixels in the original image whose value is above the threshold level. If the pixel value is below the threshold, the pixel value 0 is assigned. The values do not have to be strictly 0 for background and 1 for foreground objects; any two distinct values are sufficient. Figure 3.1 shows an original image and its thresholded counterpart.

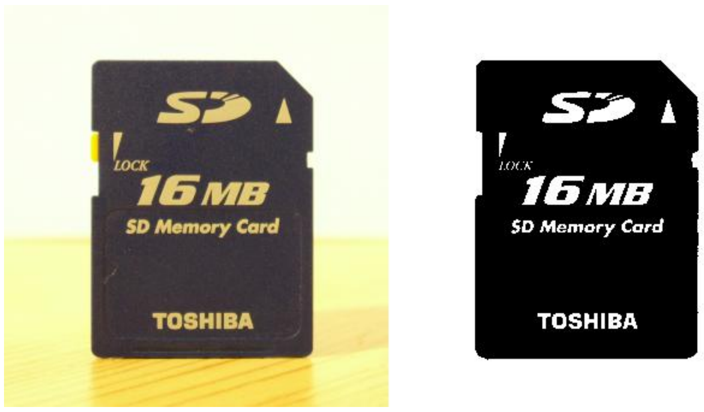


Figure 3.1: Thresholding. Left: original image; right: thresholded image.

Various methods of finding an optimal threshold exist. In the simplest case, when object and background can be clearly differentiated based on brightness level alone and the image histogram is bi-modal, the optimal threshold level lies somewhere in the valley between the two modes (peaks).

In such cases, one mode represents foreground objects and the other mode the background. However, some problems have to be dealt with when considering such an approach. In [11], the following difficulties are listed:

1. "The valley may be so broad that it is difficult to locate a significant minimum.

2. There may be a number of minima because of the type of detail in the image, and selecting the most significant one will be difficult.

3. Noise within the valley may inhibit location of the optimum position.

4. There may be no clearly visible valley in the distribution because noise may be excessive or because the background lighting may vary appreciably over the image.

5. Either of the major peaks in the histogram (usually that due to the background) may be much larger then the other, and this will than bias the position of the minimum.

6. The histogram may be inherently multimodal, making it difficult to determine which is the relevant thresholding level."

The essence of this approach is depicted in Figure 3.2.

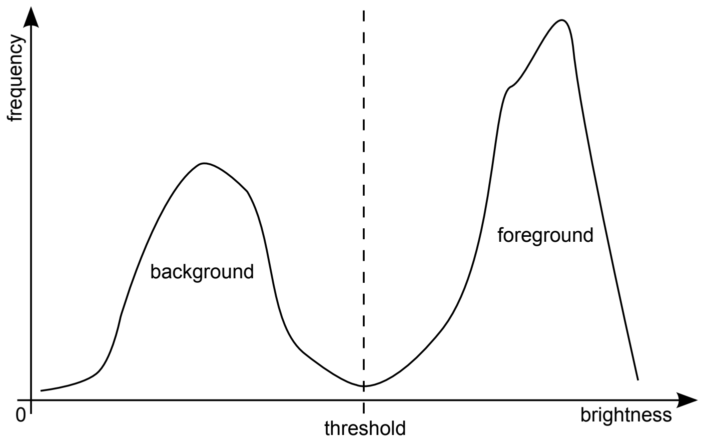


Figure 3.2: Bi-modal histogram. One mode represents dark background, the other represents a bright foreground object.

Ever since the advent of thresholding, there have been many methods devised for automatic threshold determination. One of these is a variance-based technique also known as Otsu's method [21], which will now be described. In the variance-based approach, the optimal threshold level is selected based on the ratio of between-class variance and total variance. First, assume an image with L gray levels. Let n_i denote the number of pixels with gray level i. The total number of pixels in the image is therefore $N = n_1 + n_2 + \ldots + n_L$. The probability of a pixel having a gray level i is

$$p_i = \frac{n_i}{N}, \tag{3.2}$$

where

$$p_i \geq 0, \qquad \sum_{i=1}^{L} p_i = 1. \tag{3.3}$$

Given a threshold level k, the image histogram can be divided into two parts (classes). One class is formed by the gray levels lower than k and the other by the gray levels above k. Between-class variance σ_B^2 and total variance σ_T^2 can now be computed as

$$\sigma_B^2 = \pi_0(\mu_0 - \mu_T)^2 + \pi_1(\mu_1 - \mu_T)^2, \tag{3.4}$$

$$\sigma_T^2 = \sum_{i=1}^{L}(i - \mu_T)^2 p_i, \tag{3.5}$$

where

$$\pi_0 = \sum_{i=1}^{k} p_i, \quad \pi_1 = \sum_{i=k+1}^{L} p_i, \tag{3.6}$$

$$\mu_0 = \frac{1}{\pi_0}\sum_{i=1}^{k} ip_i, \quad \mu_1 = \frac{1}{\pi_1}\sum_{i=k+1}^{L} ip_i, \quad \mu_T = \sum_{i=1}^{L} ip_i. \tag{3.7}$$

Using the aforementioned formulas, between-class variance simplifies to

$$\sigma_B^2 = \pi_0\pi_1(\mu_1 - \mu_0)^2. \tag{3.8}$$

The criterion that is to be maximized is given by the ratio of between-class variance and total variance

$$\eta = \frac{\sigma_B^2}{\sigma_T^2}. \tag{3.9}$$

As the image histogram is fixed, total variance σ_T^2 is constant, and so, to obtain the optimal threshold, only between-class variance σ_B^2 needs to be maximized. Gray level k, for which the maximal σ_B^2 is attained, is the resulting optimal threshold level.

Other methods for finding the optimal threshold level exist, such as entropy-based methods, where the thresholding level is determined as the one that maximizes total entropy of the image histogram. As in the previous case, the thresholding level divides the intensity histogram, giving rise to two histograms (foreground and background). Total entropy is then computed as the sum of the individual histogram entropies. The optimal threshold is the one for which the total entropy is maximized. The logic behind this approach is the following. Since entropy is defined as a measure of disorder in the system, maximizing total entropy of the image histogram has the effect of maximum reduction in image entropy after thresholding is performed, and thereby imposes maximum order on the system.

To sum up, image thresholding is the earliest approach to image segmentation and it is therefore no surprise that it is also the easiest one. The main advantage of thresholding is the fact that it is not computationally demanding, unlike other segmentation algorithms. This, however, goes hand in hand with the limited usability, where often rather strict conditions on the scene illumination have to be met to allow for meaningful results to be obtained.

3.2 Split-and-Merge Methods

Split-and-merge methods fall into a broader category of region-based image segmentation methods. It is convenient to first briefly introduce the bottom-up region merging approach and the top-down region splitting approach before split-and-merge methods are described.

Bottom-up region merging methods start with an image that is initially divided into regions that coincide with individual pixels. Segments are formed by successively merging smaller regions into larger ones. For merging to occur, certain homogeneity measure requirements have to be met. For example, the measure could be defined as the difference in the intensity level of individual regions. In case the difference in intensity is higher than the prescribed tolerance, regions are not merged and vice versa. Different homogeneity measures can take into account information about region texture, color and so on. The merging ends when there are no two regions that can be merged.

Top-down region splitting methods involve, as the name suggests, successive splitting of the initial region corresponding to the whole image. Splitting of a region into four quadrants occurs when the region's homogeneity tolerance is not satisfied and vice versa.

The authors of [24] mention the fact that region splitting methods can produce different segmentation results than region merging methods even if the same homogeneity criteria are used, and hence conclude that these two approaches are not dual.

Now that the workings of region splitting and merging methods have been outlined, a technique combining the advantages of both approaches called split-and-merge can be described. The need for this approach arose from one unpleasant feature of the previously described methods. As Haralick [15] notes, "Because segments are successively divided into quarters, the boundaries produced by the split technique tend to be squarish and slightly artificial. Sometimes adjacent quarters coming from adjacent split segments need to be joined rather than remain separate." For this reason, Horowitz and Pavlidis [16] proposed a split-and-merge method, which takes as its initial starting point arbitrary intermediate image partitioning and recursively proceeds in both directions (splitting as well as merging). Among the advantages of this approach, improved segmentation quality without increased memory requirements and improvement of computational speed can be counted.

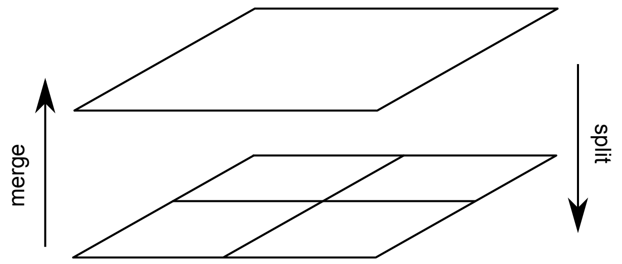


Figure 3.3: Split and merge.

For their operation, split-and-merge methods use pyramidal image representation, in which regions are square-shaped. If four regions with approximately same the homogeneity measure are found on any level of the pyramid, they are merged together to form a region on a higher level of the pyramid. Conversely, if any region on any level of the pyramid is not homogeneous enough, it is split into four subregions.

In terms of computational complexity, region-growing methods are considered to be computationally expensive and thus not well-suited for industrial applications [3]. However, in [16], the authors claim that savings in computation can be achieved by appropriate choice of initial intermediate level of image partitioning.

3.3 Clustering

Another approach to image segmentation is clustering. From the perspective of machine learning, clustering falls into the category of unsupervised learning. In unsupervised learning, the dataset of n vectors $D = \{\mathbf{x}^i\}_{i=1}^n$ is given and the goal is to discover, which vectors are similar and therefore naturally form clusters. The term ”unsupervised” refers to the fact that individual vectors in the dataset are not labeled (i.e. for each vector there is no additional information on whether the particular vector belongs to one cluster or another).

Taken from the perspective of computer vision applications, clustering makes use of the fact that groups of similar pixel brightness values in the image naturally form clusters. Instead of working only with one-dimensional information (like pixel brightness), it is also possible to take into account the pixel neighborhood. In such a case, for each pixel in the image, an m-dimensional feature vector $\mathbf{x} = [x_1, \dots, x_m]$ is formed describing the pixel in a wider context and so enabling, for example, extraction of textural information.

An essential part of a clustering algorithm is the ability to compare two feature vectors; that is, to determine how far one vector is from another. For that purpose, a number of vector space metrics can be used. Let $\mathbf{x}^i, \mathbf{x}^j \in \mathbb{R}^m$ be two feature vectors describing local properties in the neighborhood of i-th and j-th image pixels. The metric defined as

$$\rho(\mathbf{x}^i, \mathbf{x}^j) = \|\mathbf{x}^i - \mathbf{x}^j\|_2, \tag{3.10}$$

is called the *Euclidean metric*, where

$$\|\mathbf{x}^i - \mathbf{x}^j\|_2 = \sqrt{\sum_{k=1}^{m} (x_k^i - x_k^j)^2} \tag{3.11}$$

is an ℓ^2-norm. In such a case, it is said that the metric is induced by a norm. Aside from the Euclidean metric, a wide array of other metrics is available. For example, the Euclidean metric is in fact a special case of a metric induced by an ℓ^p-norm, given by

$$\sqrt[p]{\sum_{k=1}^{m} (x_k^i - x_k^j)^p}, \tag{3.12}$$

where $1 \leq p < \infty$ is the parameter. For instance, when $p = 1$, a Manhattan distance is obtained. When $p = 2$, the Euclidean metric is acquired. Every metric ρ must also satisfy the following conditions:

$$\text{non-negativity} : \rho(\mathbf{x}, \mathbf{y}) \geq 0, \tag{3.13}$$

$$\text{identity} : \rho(\mathbf{x}, \mathbf{y}) = 0 \iff \mathbf{x} = \mathbf{y} \tag{3.14}$$

$$\text{symmetry} : \rho(\mathbf{x}, \mathbf{y}) = \rho(\mathbf{y}, \mathbf{x}), \tag{3.15}$$

$$\text{triangle inequality} : \rho(\mathbf{x}, \mathbf{z}) \leq \rho(\mathbf{x}, \mathbf{y}) + \rho(\mathbf{y}, \mathbf{z}), \tag{3.16}$$

where $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^p$. With the notion of a metric outlined, we can now proceed to explain the simplest clustering algorithm.

One of the earliest clustering algorithms is the k-means algorithm. The principle objective of k-means clustering lies in minimization of within-cluster squared distance. Let $\{\mathbf{x}_i\}_{i=1}^n$ be a set of n feature vectors that we wish to partition into a set of k mutually exclusive and exhaustive clusters $\mathbf{C} = \{C_1, \ldots, C_k\}$. The goal of k-means clustering is to find partitioning that minimizes

$$\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in C_i} \|\mathbf{x}_j - \mu_i\|^2, \tag{3.17}$$

where $\mathbf{S}$ is set of all possible divisions of the dataset and μ_i denotes the mean of the i-th cluster, also called cluster *centroid.* The following points summarize the algorithm.

K-means clustering:

1. Randomly pick k initial vectors $\mathbf{x}_1, \ldots, \mathbf{x}_k$ as cluster centroids.
2. For each vector $\mathbf{x}_i$, determine its distance to each of the k centroids (evaluate metric) and assign $\mathbf{x}_i$ to the cluster represented by the closest centroid.
3. Recalculate cluster centroids (compute the mean of the clusters).
4. Repeat steps 2 and 3 until convergence is achieved (cluster centroids cease to change or only change very slightly).

The algorithm is guaranteed to converge to a solution, which does not have to be necessarily optimal. Results of the k-means clustering algorithm are highly dependent on the way the initial centroids are chosen and the number of clusters k. For that reason, it is usually recommended to run the algorithm multiple times, with the initial centroids being randomly chosen on every run.

The result of the application of k-means clustering to color image segmentation is displayed in Figure 3.4. More in-depth discussion of various cluster-

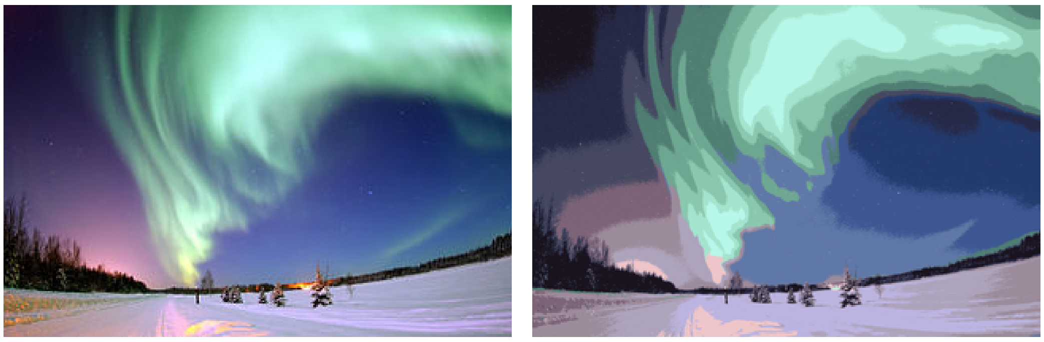


Figure 3.4: Original image on the left and segmented image using a k-means ($k = 16$) clustering algorithm on the right.

ing algorithms with applications to image segmentation can be found in [19]. An overview of contemporary clustering techniques is discussed in [20].

3.4 Classification

In the previous section, unsupervised learning was outlined as one possible way to image segmentation. Unsupervised learning is characterized by the fact that the data being processed lack ground-truth labels. Opposed to this is a supervised learning approach, where individual vectors are assigned a label.

In supervised learning, a dataset $D = \{(\mathbf{x}^i, y^i)\}_{i=1}^n$ is given, where a pair $(\mathbf{x}^i, y^i)$ is called the *training example*, which consists of p-dimensional *feature vector* $\mathbf{x}^i \in \mathbb{R}^p$ and corresponding *label* $y^i \in \mathbb{R}$. The goal is to learn a predictor $y^* = g(\mathbf{x}^*; \theta)$ from available training data D, which will accurately predict the label value y^* for the new previously unseen data vector $\mathbf{x}^*$. In case the predicted label value y^* is discrete, this is called a *classification problem*, if the predicted value y^* is continuous, this is referred to as a *regression problem* [4]. For the purposes of this thesis, only the classification problem will be further considered.

The learning algorithm which performs the task of classification is called the *classifier*. Each classifier operates in two phases. During the *training phase*, available data are used to yield an optimal estimate θ^* of the classifier's parameter. This process is known as *training*. In the *application phase*, the optimal parameter of the classifier is set. Previously unseen data are presented to the classifier and class label predictions are obtained for each new input vector.

When considering the classification of objects, it is of paramount importance to identify that which differentiates a particular class of objects from another class. Classification performance is greatly affected by the choice of measured features, which form the feature vector. To give an example, height and weight are good features to measure when detecting whether a particular person is overweight or not. On the contrary, the color of one's skin or size of one's feet have no bearing on one's excessive weight. These are examples of features that have no discriminatory value, and thus do not contribute to the performance of the classifier.

Another factor affecting the performance of the classifier is the problem of high *bias* and high *variance*. When the classifier suffers from high bias, the trained model is of too low complexity to accurately discriminate the individual classes. This is also referred to as the problem of *underfitting*. On the other hand, when the classifier suffers from a high variance problem, the trained model is of higher complexity than is necessary, which is also referred to as the problem of *overfitting*. A classifier that has just the right complexity (it doesn't underfit nor overfit the data) is said to *generalize* well.

To determine which of these problems lies behind the poor performance of the classifier, it is helpful to first divide the dataset D into two parts. One part of the dataset, called the training set, is used for training of the classifier; the second part, the test set, represents unseen data and is used for the final performance evaluation. Figure 3.5 shows the error of the training set as a function of model complexity. The optimal model complexity can be

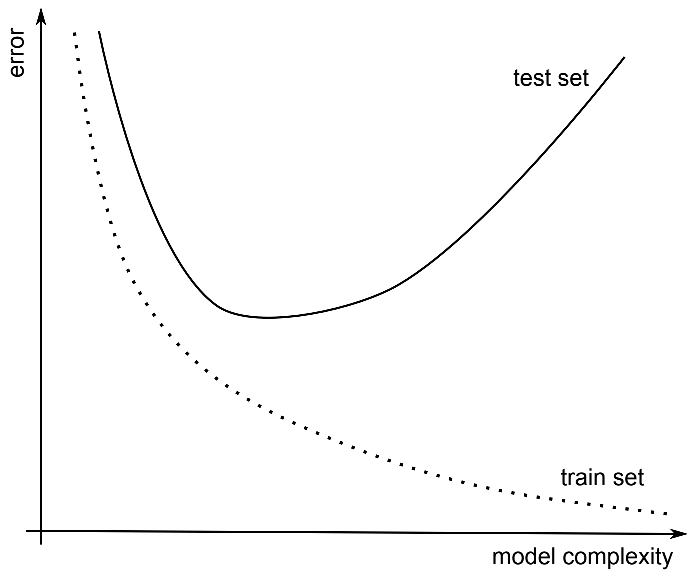


Figure 3.5: Training and test set errors as a function of model complexity.

determined by finding the minimum of test set error.

To prevent the problem of overfitting (or underfitting), some algorithms employ a *regularization* parameter, which penalizes the model flexibility and, if set correctly, ultimately prevents the aforementioned problems. In this case, it is often common practice to divide the dataset into three parts: training set, validation set and test set. As in the former case, the training set is used to train the classifier and the test set is used for the final performance

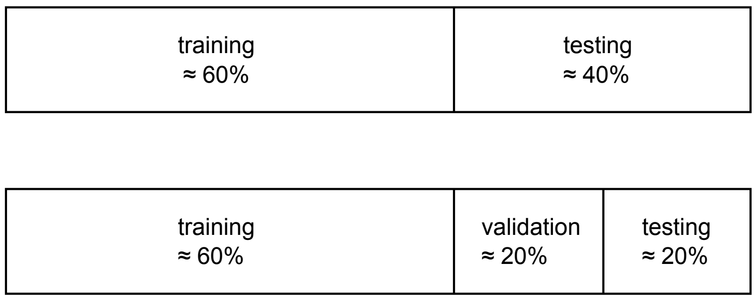


Figure 3.6: Possible division of dataset for reasons of performance evaluation and regularization parameter fine-tuning.

assessment. Before the final performance can be assessed, the validation set is used to find the optimal setting of the regularization parameter, which ensures the optimal bias/variance trade-off. As shown in Figure 3.7, error on the training set and error on the validation set are plotted as a function of regularization parameter λ. When λ is low, the model is more flexible

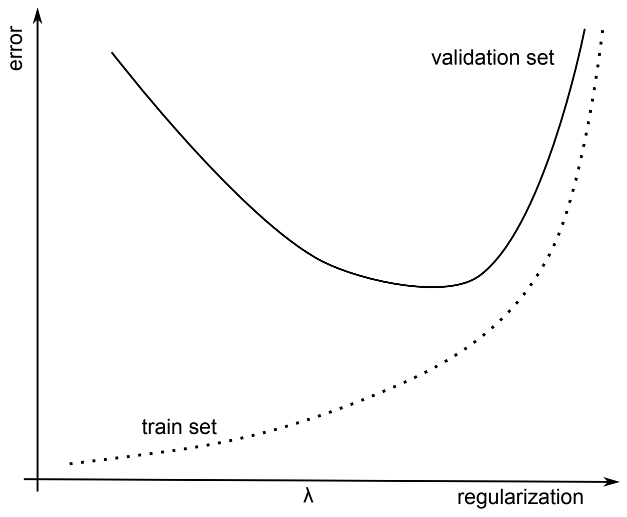


Figure 3.7: Training and validation set errors as a function of regularization parameter λ.

and therefore the error on the training set is low. However, the validation set error is high, because during training, the model relied too much on the training data and when presented with new previously unseen data it makes a lot of errors. With more regularization applied (increased λ), the model flexibility becomes more restricted and as a result the training set error gradually increases. As the validation set error develops for increasing λ, at a certain point it reaches minimum and then starts rising again. It is at this point where regularization strength λ is deemed to have the optimal value.

When solving the problem of image segmentation by supervised learning, a classifier is trained to classify each pixel as belonging to either foreground or background. First, a suitable set of images that are representative of the task being solved needs to be selected. Having chosen sufficiently discriminatory features, a feature extraction procedure can form a feature vector for each pixel in the image. In addition to that, the user needs to manually label every pixel. Labeling can be very tedious and time-consuming and for some classification problems, even a very expensive process. After the dataset is prepared, the classifier can be trained and tested. Once we are confident that the classifier can produce sufficiently accurate predictions, it is finally ready to perform the task of image segmentation.

Over the years of research in machine learning and related fields, a great number of learning algorithms have been devised. As a result, the choices of classification algorithms at present are numerous. As a good reference for machine learning algorithms, [4, 5, 12, 20] are highly recommended. The SVM classifier that was chosen as suitable for the purposes of this thesis, is described in the following chapter.

Chapter 4

Support Vector Machines

Support vector machines are supervised learning models that can be used for classification as well as regression. In this thesis, attention is directed only to the support vector classification. Throughout this chapter we will use the abbreviation SVM in the sense of support vector classification.

A support vector machine is a binary linear classifier that arose from the results of statistical learning theory (also called Vapnik-Chervonenkis theory). It was first proposed by V. N. Vapnik in 1979 [28]. An SVM is sometimes said to belong to the class of decision machines [5], since it only decides to which class a given object belongs and does not provide a posterior probability of the predictions. The great advantage of SVMs is that they are trading-off bias and variance, thus preventing overfitting and ensuring good generalization performance. Furthermore, the problem can be formulated as a convex optimization problem, whose solution is unique (follows from convexity) and also sparse. Sparsity of the solution is a very desirable property when computational aspects come into play.

To illustrate the development in the field of support vector machines, we will gradually work our way to the final form of the SVM classifier as it is used today.

4.1 Optimal Separating Hyperplane

Before we start to talk about SVMs, it is helpful to introduce the notion of an optimal separating hyperplane. We will assume a set of n training examples $(\mathbf{x}_i, y_i)$ forming a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^p, y_i \in \{-1, 1\}\}_{i=1}^n$, where $\mathbf{x}_i$ is the input *feature vector* and variable y_i is the *label*, which encodes class membership of $\mathbf{x}_i$. The dataset is said to be *linearly separable* if all the feature vectors with label $y_i = 1$ can be separated from all the feature vectors with label $y_i = -1$ by a $(p-1)$-dimensional linear hyperplane in $\mathbb{R}^p$. Since an SVM is a linear classifier the discriminant function has the form

$$h(\mathbf{x}) = \mathbf{w}\mathbf{x} + b, \tag{4.1}$$

where $\mathbf{w}$ is the normal vector perpendicular to the hyperplane and b is the bias term.

The problem we are facing is the following; given a linearly separable dataset of training examples $\mathcal{D}$, find a linear hyperplane among all possible linear hyperplanes that separates the data with the largest margin. Roughly speaking, a hyperplane separates data with the largest margin if the distance from the hyperplane to the nearest points of both classes is maximal. It can be shown with the use of VC theory [29] that maximizing the margin is one of the ways to achieve optimal generalization performance of the classifier.

Returning back to the fact of multiple possible separating hyperplanes, consider Figure 4.1 for illustration. There are many linear hyperplanes that are capable of separating the dataset; however, only one of them separates the data with the largest margin. In order to determine which of the many possible linear hyperplanes is the optimal one, the problem will be formulated as a convex optimization problem with linear constraints, whose unique solution is value $\mathbf{w}^*$ defining the hyperplane that we seek. Finally, the class membership of the new previously unseen feature vector $\mathbf{x}$ will be assigned according to the decision rule

$$g(\mathbf{x}) = \operatorname{sign}\left(\mathbf{w}^*\mathbf{x} + b\right). \tag{4.2}$$

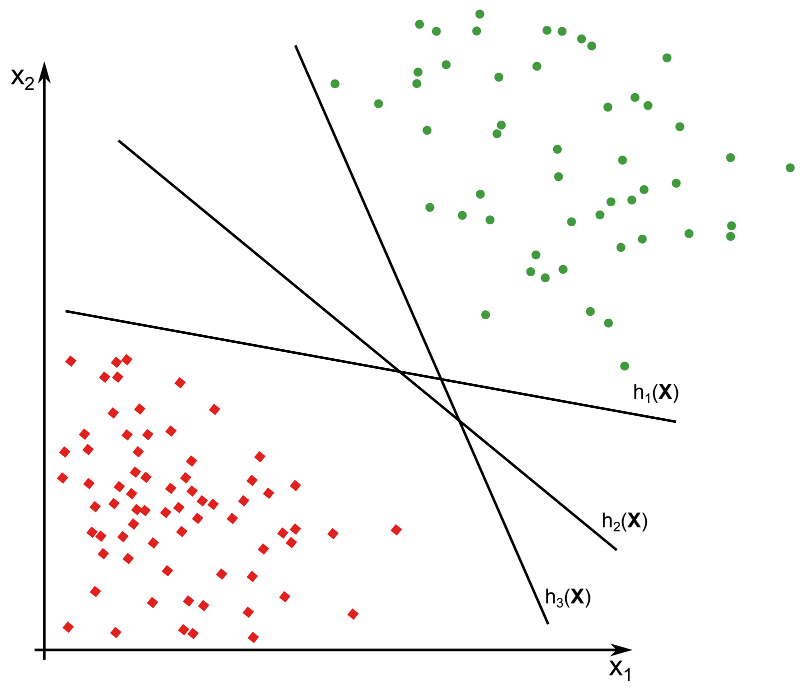


Figure 4.1: There are many ways in which the two classes can be separated.

4.2 Hard Margin

First, we will consider a case where the data is linearly separable. The hyperplane is thus assumed in the form of (4.1). As mentioned before, we are seeking a hyperplane that separates the data with the largest possible margin d, as depicted in Figure 4.2. To express the margin in terms of the parameters of the hyperplane $\mathbf{w}$ and b, consider two vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, which lie exactly at the bounds. It holds

$$\mathbf{w}\mathbf{x}_1 + b = 1, \tag{4.3}$$

$$\mathbf{w}\mathbf{x}_2 + b = -1. \tag{4.4}$$

Subtracting (4.4) from (4.3) and normalizing $\mathbf{w}$ to unit length, we obtain

$$\frac{\mathbf{w}}{\|\mathbf{w}\|^2} \cdot (\mathbf{x}_1 - \mathbf{x}_2) = \frac{2}{\|\mathbf{w}\|^2}, \tag{4.5}$$

where $\|\mathbf{w}\|^2 = \mathbf{w}^{\mathrm{T}}\mathbf{w}$ is a squared Euclidean norm. The margin is thus given by a projection of $(\mathbf{x}_1 - \mathbf{x}_2)$ onto the normal vector $\mathbf{w}$ and is equal

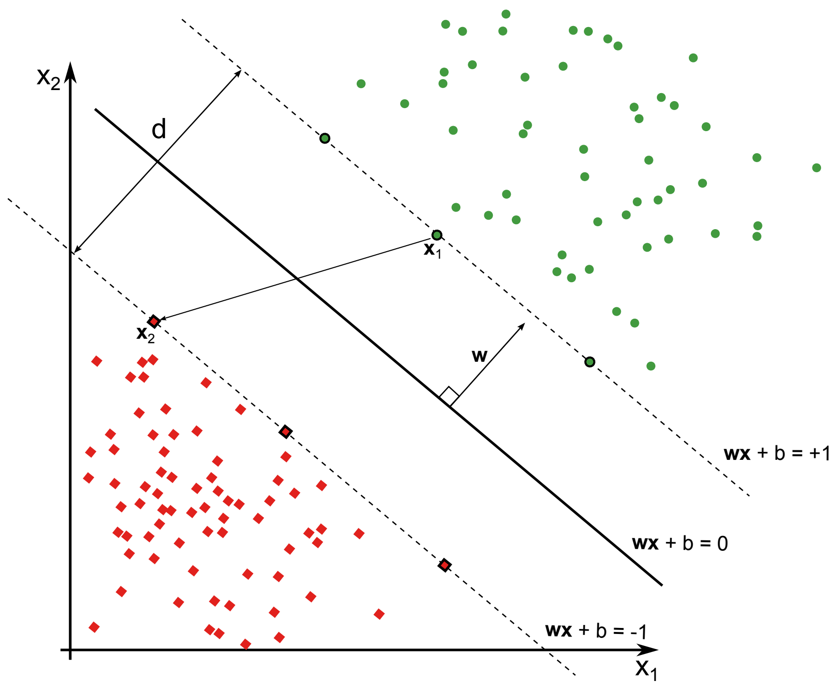


Figure 4.2: Hard margin; no error in classification is allowed.

to $d = 1/\|\mathbf{w}\|^2$. In order to maximize d, the quantity $\|\mathbf{w}\|^2$ must be minimized, and so the optimization problem becomes

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}, \tag{4.6}$$

subject to constraints

$$y_i \cdot (\mathbf{w}\mathbf{x}_i + b) \geq 1, \qquad i = 1, \ldots, n. \tag{4.7}$$

The objective function is quadratic and the constraints are linear; this problem is therefore one of quadratic programming. To solve this, we use the method of Lagrange multipliers, which converts the original constrained optimization problem into an unconstrained one. The Lagrange function is

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} - \sum_{i=1}^{n} \alpha_i [y_i(\mathbf{w}\mathbf{x}_i + b) - 1], \tag{4.8}$$

where α_i are Lagrange multipliers. Taking partial derivatives

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = 0 \Leftrightarrow \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i, \tag{4.9}$$

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial b} = 0 \Leftrightarrow 0 = \sum_{i=1}^{n} \alpha_i y_i, \tag{4.10}$$

and substituting back into (4.8), we obtain the Wolfe dual

$$W(\alpha) = \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j). \tag{4.11}$$

The dual must be maximized

$$\max_{\alpha} W(\alpha) \tag{4.12}$$

under the constraints

$$0 \leq \alpha_i, \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \qquad i = 1, \ldots, n. \tag{4.13}$$

The expression for the final resulting separating hyperplane

$$g(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + b \right). \tag{4.14}$$

can be obtained by plugging (4.9) into (4.1).

Because of the fact that the obtained solution is sparse, some of the coefficients α_i will be zero. As a result of that, the optimal separating hyperplane is determined only by feature vectors $\mathbf{x}_i$ for which $\alpha_i > 0$. These are called *support vectors*.

4.3 Soft Margin

In real-world applications, linearly separable classes are more of a rarity than a common occurrence. We are thus more likely to encounter a problem where the classes are overlapping. This can be caused either by the nature of the data itself (presence of outliers) or simply by the inevitable presence of noise

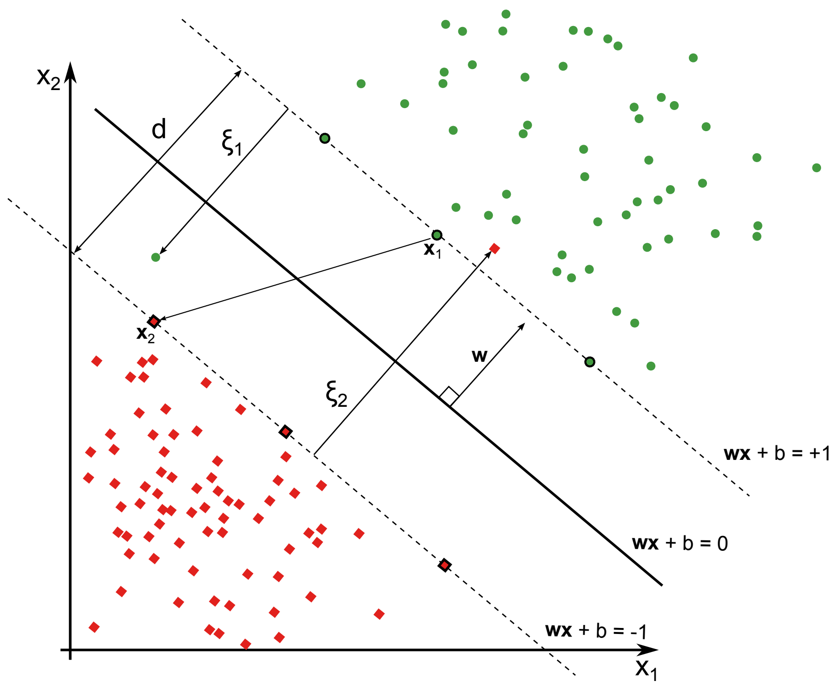


Figure 4.3: Soft-margin; misclassification is allowed to a certain degree.

in measurements. In 1995, Cortes and Vapnik [9] suggested a modification of the idea of a maximum margin hyperplane that can handle the case of overlapping classes.

The situation is captured in Figure 4.3. To solve this problem, *slack variables* were introduced into the inequality constraints. Slack variables ξ_i measure the degree of misclassification of $\mathbf{x}_i$, therefore relaxing the constraints and consequently making the model more flexible. In order for the misclassification to be minimal, slack variables need to be added to the objective function. We have

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} + C\sum_{i=1}^{n}\xi_i, \tag{4.15}$$

subject to constraints

$$y_i \cdot (\mathbf{w}\mathbf{x}_i + b) \geq 1 - \xi_i, \qquad i = 1, \ldots, n. \tag{4.16}$$

The parameter $C > 0$ is called the *regularization constant*, which "controls the trade-off between slack variable penalty and margin" [5]. The Lagrangian is

$$L(\mathbf{w}, b, \alpha, \xi) = \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i[y_i(\mathbf{w}\mathbf{x}_i + b) - 1] - \sum_{i=1}^{n}\beta_i\xi_i, \quad (4.17)$$

where $\alpha_i, \beta_i \geq 0, \quad i = 1, \ldots, n$ are Lagrange multipliers. Taking partial derivatives

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \Leftrightarrow \mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i, \quad (4.18)$$

$$\frac{\partial L}{\partial b} = 0 \Leftrightarrow 0 = \sum_{i=1}^{n}\alpha_i y_i, \quad (4.19)$$

$$\frac{\partial L}{\partial \xi_i} = 0 \Leftrightarrow C - \alpha_i - \beta_i = 0 \quad (4.20)$$

and substituting back into (4.17), we obtain the same dual as before

$$W(\alpha) = \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i \cdot \mathbf{x}_j). \quad (4.21)$$

The dual must be maximized

$$\max_{\alpha} W(\alpha) \quad (4.22)$$

subject to constraints

$$0 \leq \alpha_i \leq C, \quad \sum_{i=1}^{n}\alpha_i y_i = 0, \qquad i = 1, \ldots, n. \quad (4.23)$$

The optimal value of regularization parameter C must be found experimentally using a validation set.

On a side note, there exists yet another approach that incorporates slack variables. What was shown above is the approach of minimizing an L_1-norm; however, an L_2-norm minimization is also possible. This is done by replacing the regularization term $\sum_i \xi_i$ in the objective function by $\sum_i {\xi_i}^2$.

4.4 Kernel Trick

In previous sections we have dealt with two important cases. Namely, the linearly separable case, the least complicated of all, and the case of overlapping classes, where the overlap may be due to inherent noise in feature measurement. However, so far we have not addressed the most common scenario often encountered in many real-world applications: the case of linearly non-separable classes.

One of the things that makes SVMs so powerful is the so-called *kernel trick*, which is used throughout machine learning (not only in SVMs) and is very helpful in dealing with this problem. It was first introduced by Aizerman

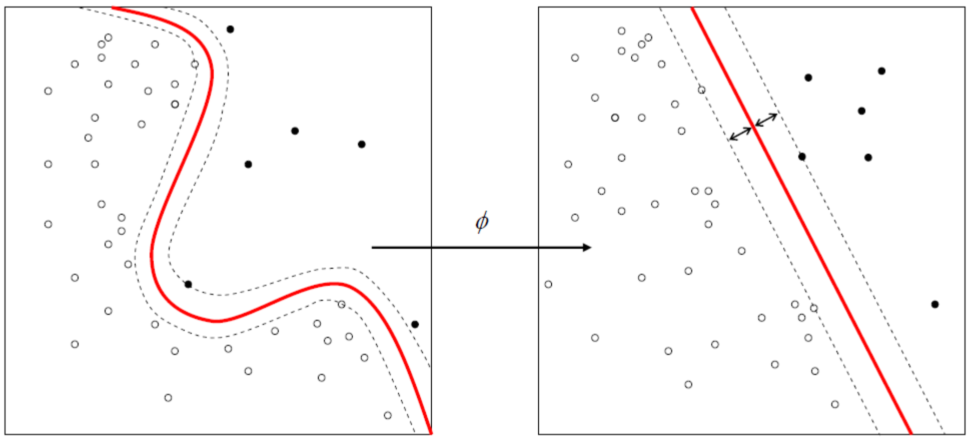


Figure 4.4: Kernel trick: On the left we see the original input space; the separating hyperplane here is nonlinear. On the right is transformed higher dimensional feature space, where data are linearly separable.

et al. in 1964 [1]. The main idea of the kernel trick is to create an alternative representation of the data in higher-dimensional feature space, where the data are linearly separable, by means of a nonlinear transformation.

Let us direct our attention to the dual form of the optimization problem

$$W(\alpha) = \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j); \tag{4.24}$$

notice the dot product term in double summation. We want to transform data points into higher-dimensional feature space, and for that we introduce the so-called *kernel function*

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j). \tag{4.25}$$

By defining a kernel, we are implicitly defining a transformation ϕ, which doesn't have to be known explicitly. Evaluation of the kernel for any two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ will give us a representation of the dot product $\mathbf{x}_i \cdot \mathbf{x}_j$ in transformed feature space. The dual objective (4.24) can therefore be modified accordingly

$$W(\alpha) = \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \cdot k(\mathbf{x}_i, \mathbf{x}_j), \tag{4.26}$$

resulting in a separating hyperplane given by

$$g(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_i y_i \cdot k(\mathbf{x}_i, \mathbf{x}) + b \right). \tag{4.27}$$

A useful fact can be noted here. The solution α is sparse, which means that a good amount of α_i will be zero. The summation in (4.27) will therefore concern only support vectors, since for support vectors it holds that $\alpha_i > 0$.

For a function to qualify as a kernel it needs to meet certain requirements, which are extensively discussed in detail in [22] and also [10]. We will only mention examples of popular choices of kernels that are widely used. These are

$$\begin{aligned} \text{linear} : k(\mathbf{x}_i, \mathbf{x}_j) &= \mathbf{x}_i \cdot \mathbf{x}_j, & (4.28)\\ d\text{-th degree polynomial} : k(\mathbf{x}_i, \mathbf{x}_j) &= (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d, & (4.29)\\ \text{radial basis function} : k(\mathbf{x}_i, \mathbf{x}_j) &= \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), & (4.30)\\ \text{sigmoid} : k(\mathbf{x}_i, \mathbf{x}_j) &= \tanh(\gamma(\mathbf{x}_i \cdot \mathbf{x}_i) + r). & (4.31) \end{aligned}$$

A kernel doesn't only have to be defined by a function. It can also be defined by an algorithm. This allows for kernels to be defined for all kinds of objects like graphs, strings, etc. Examples of such kernels can be found in [10] and [20].

Chapter 5

Feature Extraction

A vital part of every automatic Vickers indentation inspection system is obtaining measurements of the impression in the material. To achieve this goal, some solutions resort to detection of corners of the square-shaped impression, while others try to segment out the inner region of the impression. In this thesis, the supervised learning approach to image segmentation was chosen. To accomplish this task, the SVM classifier was the natural choice because of its robustness to noise.

Having settled on which classifier to use, the next crucial step is, obviously, the design and choice of appropriate features and assembling them into a feature vector. In this chapter, a process of feature extraction, which serves as a preliminary step to SVM learning, is described in detail.

5.1 Data and Labeling

An optical system that captures the indentation in the polymer produces a series of images with 1280×1024 resolution. The images used for this thesis were provided in PNG format, every image approximately 700kb in size. The series of 50 images depicts an impression in PMMA (plexiglass) polymer as it shrinks in time. Roughly 10 initial images in the series had to be thrown away due to overexposure.

Because the chosen approach to image segmentation is supervised learning, a training set had to be created. In order for the sequence to be represented as a whole, the training set was formed by selecting images from the whole series. Copies of these images were labeled. Labeling consisted of loading the image in bitmap editor GIMP, carefully outlining the impression in the polymer by hand and then using a bucket-fill tool to fill the inside of the selection. It must be noted that the process of labeling is highly subjective, because the edges of the impression in the image were blurry and hardly discernible, thus presenting difficulties in determining where the edge truly lies, especially when zoomed in on the detail. Figure 5.1 shows an example of the original provided PNG image and the labeled copy of the same image.

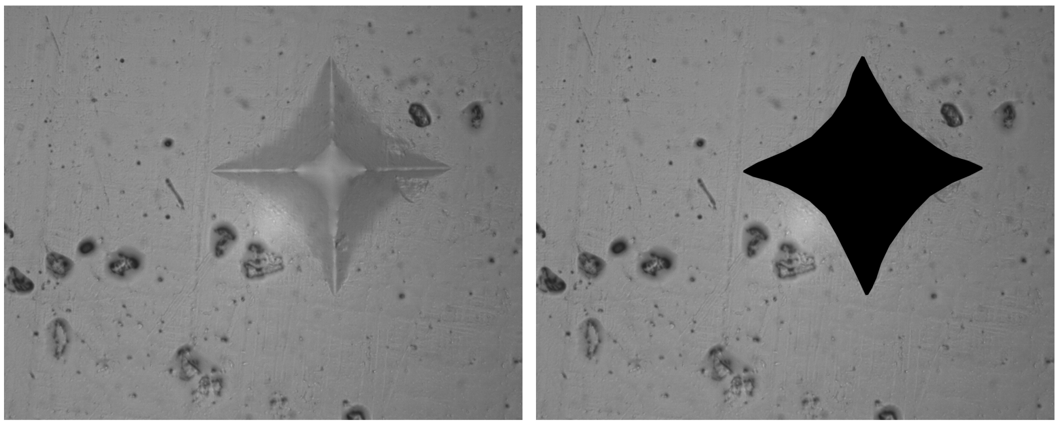


Figure 5.1: Provided image in full resolution and its labeled counterpart.

5.2 Preprocessing Operations

Because of the inherent row noise introduced by the camera electronics in the supplied images, measures to alleviate this problem had to be taken. To remove the row noise, pixel values in every row were summed up and the resulting column vector was mean-filtered. The difference between the unfiltered and filtered column vector of row sums was calculated. Difference obtained was normalized (by the number of pixels in the column vector, i.e. image height) and subsequently subtracted from every column of the image. Additional to that, the whole image was then filtered with a low-pass frequency filter. The result of the noise removal process can be seen in Figure 5.2.

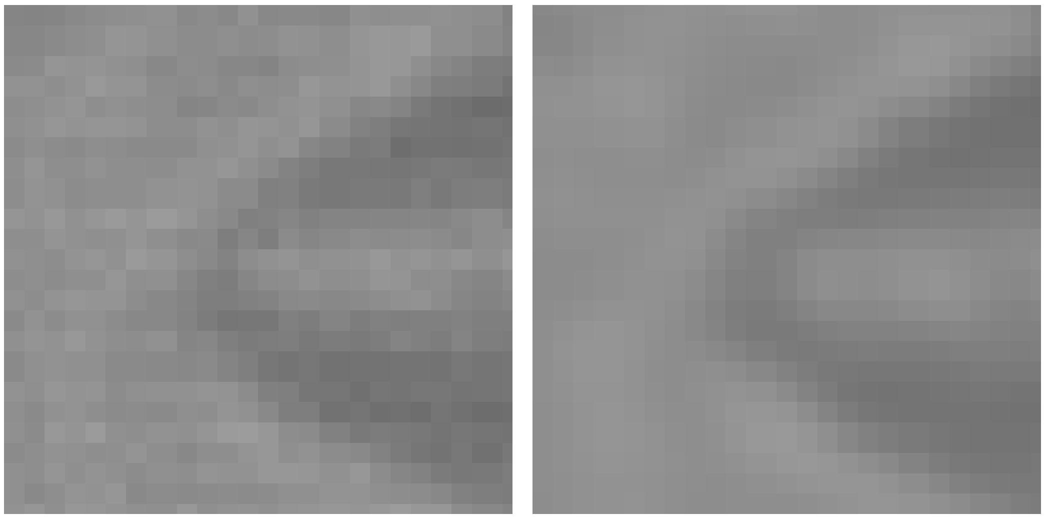


Figure 5.2: Detail of original unfiltered image (left) and image after noise removal (right).

As the SVM classifier is not the algorithm that scales best with the amount of data, considerations that would reduce the computational load had to be taken into account. If a feature vector would be formed for every pixel of the 1280×1024 image, this would result in $1280 \cdot 1024 = 1,310,280$ feature vectors for every image in the sequence, which is simply intractable. For this reason, every image was resized by a factor of 0.4.

Furthermore, to ensure that the two classes of pixels (background and foreground) are equally represented, a crop that contained the impression was necessary. This was done also for the purpose of the final performance evaluation. Should the foreground and background be unequally represented, this would introduce the problem of skewed classes and certain classifier performance metrics would be inadequate.

Finally, as the images suffer from low contrast, a contrast enhancement was required. At first, ordinary histogram equalization was considered. However, because the illumination in the images is uneven, with a halo appearing in the lower left corner, ordinary histogram equalization produced an image where the significant edges were lost in the process. Instead, an adaptive

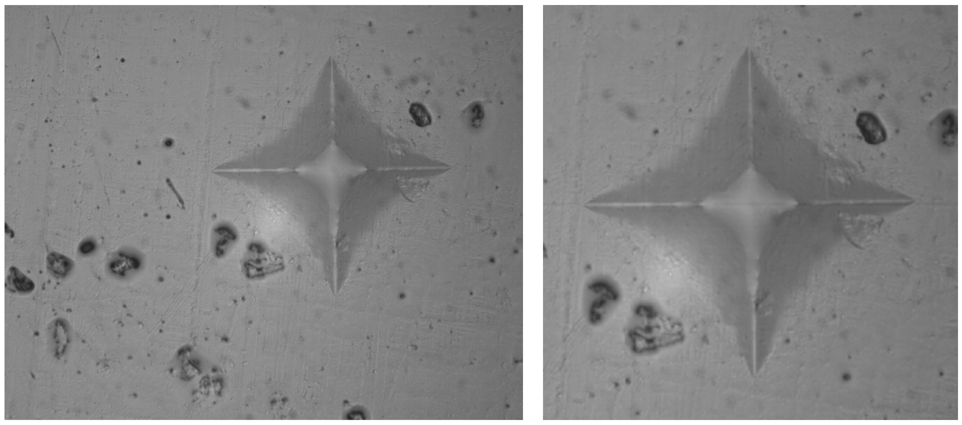


Figure 5.3: Original image (resolution: 1280 × 1024) and the image after resizing and cropping (resolution: 298 × 298).

histogram equalization, which takes into account the local image properties, was considered. Nonetheless, features extracted from these images still

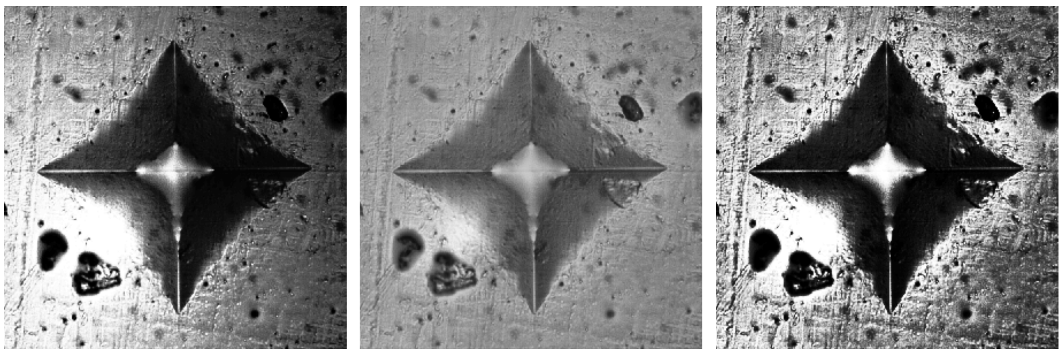


Figure 5.4: Histogram equalization, adaptive histogram equalization and a combination of the two.

yielded unsatisfactory results. When images enhanced with adaptive histogram equalization were further subjected to ordinary histogram equalization, the classification results improved. Figure 5.4 compares the effect of ordinary histogram equalization, adaptive histogram equalization and the successive application of the two.

5.3 Variance Features

As stated in the final section of the previous chapter, in order to classify each pixel as belonging to either foreground or background, a feature vector that describes the local pixel properties must be created. This feature vector must take into account the most relevant and salient features that will discriminate well between the two classes of pixels.

To this end, features based on image variance are proposed in this thesis. The reason for choosing variance is the fact that variance is a statistical measure of variability. The impression is characterized by large areas more or less uniform in brightness and therefore variance is presumed to be low. In contrast, the background is characterized by the presence of a great number of smaller dark blobs on the bright background and therefore the variance is expected to be higher.

A variance image is created by sliding a window of given size across the whole input image, where for every window position the variance of pixels inside the window is computed. Since the sliding window approach is computationally expensive, the properties of integral images were utilized. As can be noticed, the formula for variance can be expressed in terms of sums of the pixel values.

$$\begin{aligned}\sigma^2 &= \frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2 = \frac{1}{N}\sum_{i=1}^{N}x_i^2 - \frac{2\mu}{N}\sum_{i=1}^{N}x_i + \mu^2 \\ &= \frac{1}{N}\sum_{i=1}^{N}x_i^2 - \mu^2 = \frac{1}{N}\left(\sum_{i=1}^{N}x_i^2 - \frac{1}{N}\left(\sum_{i=1}^{N}x_i\right)^2\right) \end{aligned} \tag{5.1}$$

This fact is very useful, because the summation terms $\sum x_i$ and $\sum x_i^2$ can be easily precomputed and then used for variance computation as demonstrated by the formula. Concretely, computation of variance in every image window is simply a matter of four arithmetic operations.

In order to take into account variations in the image brightness on multiple scales, 15 different window sizes were utilized for variance image computation. The smallest window size used is 3. The window sizes go up by 4 all the way to 59, forming the following sequence: 3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51, 55, 59. As the different window sizes are utilized, each resulting variance image has different dimensions (the margin of $(windowsize - 1)/2$ is removed); the computed variance images were all cropped to the size of the variance image produced by the largest window. Therefore, the image resolution changed from 298×298 to 240×240 pixels, resulting in $240^2 = 57600$ feature vectors per input image. By calculating a

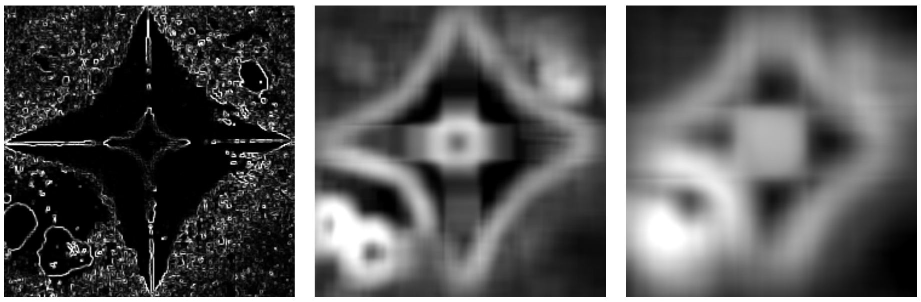


Figure 5.5: Variance images computed using window sizes 3×3, 31×31 and 59×59.

variance image using 15 different window sizes, a series of 15 variance images for every input image is obtained (Fig. 5.6). The resulting variance images can be imagined as being stacked on top of each other. The feature vector for every pixel is then constructed by selecting the corresponding pixels in every image of the series. In other words, pixels that are on top of each other form the feature vector. Because fifteen different window sizes were used during variance image computation, the resulting feature vector, describing a single pixel in a wider local context, is 15-dimensional.

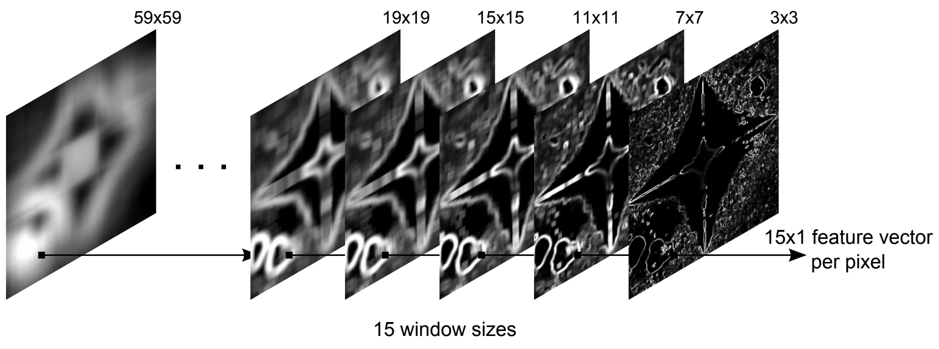


Figure 5.6: Extraction of feature vectors from a set of variance images.

Chapter 6

Experiment Design

In this chapter, binary classifier performance evaluation metrics such as precision, recall, F_1-score and Matthews Correlation Coefficient are defined and explained. After that, the design of specific experiments is justified. There were two types of experiments performed. One type of experiment is designed to evaluate the segmentation performance, with the help of performance measures mentioned above. The other type of experiment evaluates the precision with which the computer is able to locate the center of the indentation. Determining the location of the indentation center is an important initial step for other high-level feature extraction techniques.

6.1 Performance Metrics

For assessing the performance of the binary classifier, several measures can be used. Before these can be defined, it is necessary to introduce the so-called *confusion matrix*. In the case of classification into two classes (dichotomy), four distinct possible outcomes can occur. The confusion matrix is used to clearly capture the rate of these phenomena. In Figure 6.1, TP stands for *True Positive*, which, in our case, is the number of pixels that the human expert labeled as object and the classifier also assigned to the object. Correspondingly, TN stands for *True Negative*, which is the number of pixels that were labeled as background and, at the same time, predictions also indicate that they belong to the background. In other words, in both of these cases,

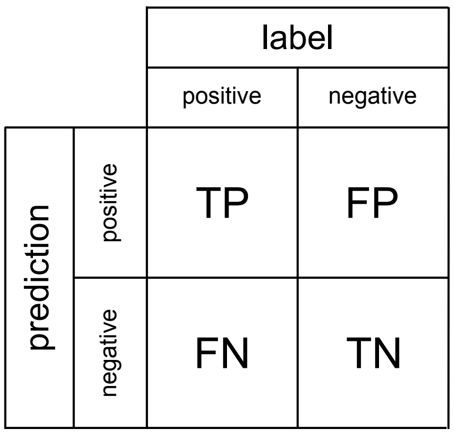


Figure 6.1: Confusion matrix. Positive represents the foreground (indentation) and negative the background.

the predicted value (determined by the classifier) matches the ground-truth label (provided by the human expert). In contrast, *False Positive* (FP) is the number of pixels that were supposed to be predicted as belonging to the background, but instead predictions say that these pixels belong to the foreground. The *False Negative* (FN) rate is defined analogously.

Having introduced the confusion matrix, several useful classifier performance measures that combine the elements of this matrix will now be defined.

Precision

Precision is the ratio of true positives to all the positive predictions (including the false positives).

$$P = \frac{TP}{TP + FP} \tag{6.1}$$

In the image segmentation context, for the precision to be high the number of pixels falsely predicted as belonging to the object (false positive rate) must be minimal. High precision can then serve as an indicator of low oversegmentation and vice versa.

Recall

Recall is defined as the ratio of true positives to all the predictions that should be predicted positive.

$$R = \frac{TP}{TP + FN} \tag{6.2}$$

In the image segmentation context, for the recall to be high the number of pixels falsely predicted as belonging to the background (false negative rate) must be minimal. High precision can then serve as an indicator of low over-segmentation and vice versa.

F_1-score

In order to combine and control the trade-off between precision and recall, the F_1-score was derived. The F_1-score is a special case of F_β-score (for $\beta = 1$), which puts weight on precision and recall equally. The formula is essentially the harmonic mean of precision and recall.

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{6.3}$$

This measure takes on values between 0 and 1, where the higher values are desirable.

Matthews Correlation Coefficient

None of the measures mentioned previously take into account the true negative rate. The consequence of this is that precision, recall and even F_1-score can provide misleading values in cases when the number of foreground pixels is significantly different from the number of background pixels (skewed/imbalanced classes). The Matthews correlation coefficient solves this problem by including a true negative rate in its formula.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{6.4}$$

Speaking in the context of image segmentation, even if the object in the image was very small (or large) in relation to the background, the Matthews correlation coefficient would still provide an unbiased performance assessment of the classifier. The coefficient takes on values in the interval $[-1, 1]$, where -1 indicates total disagreement between prediction and ground-truth label, 0 means predictions are no better than random prediction and $+1$ represents perfect prediction.

6.2 Segmentation Performance

In order to successfully track the indentation recovery in time, it is necessary to know the location of the indentation boundaries as precisely as possible. Which is why the image segmentation process has to be as accurate as possible. In the first experiment, the focus is therefore on image segmentation quality evaluation.

The twelve labeled images chosen as representatives of the image sequence were utilized as a working dataset for this experiment. The goal of this experiment is to show the learning ability of the SVM classifier for gradually increasing training set size. As we had 12 labeled images at our disposal, a decision was made to create training sets of two, four, six, eight and ten images. To put this into perspective, every 240×240 image is described by $57,600$ feature vectors. Figure 6.2 provides a diagram illustrating the ways in which the training and testing set were put together. The black dots indicate the images from the sequence that were used for training. The white dots mark the images used for testing and performance assessment. In this manner, five SVM models were trained and for each of them all the previously defined performance measures were calculated.

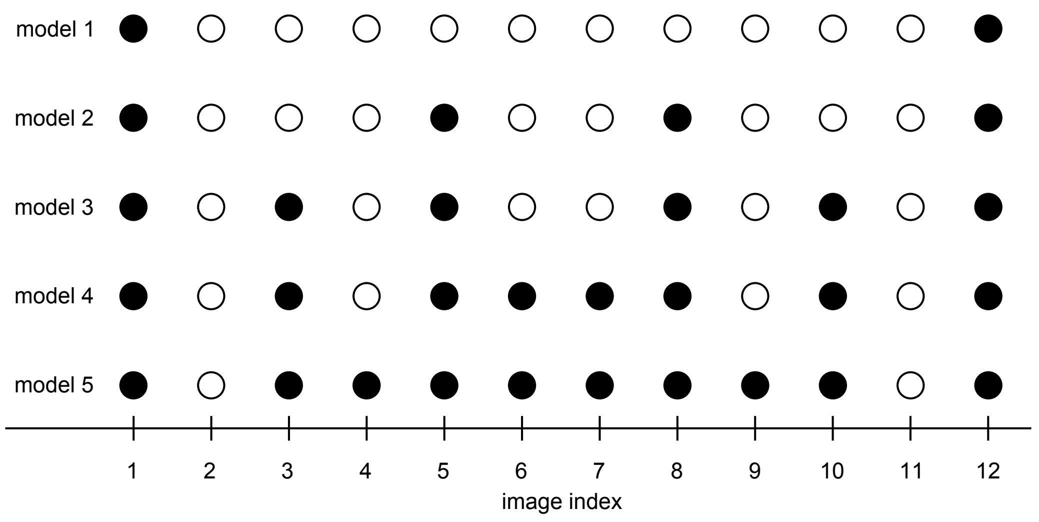


Figure 6.2: Division of the dataset for training (black) and testing (white).

Because the image segmentation process is based on per-pixel classification, the resulting predictions are expected to contain a certain amount of noise. The predictions obtained thus need to be filtered in a suitable manner. The design of the filtering step was directed by the prior information at our disposal. Namely, that the indentation is a compact object (does not contain any holes) and the fact that the indentation is the largest object in the image. In the first part of the filtering step, the holes were filled (MATLAB function `imfill()`). In the second part, only the largest connected component was selected, thus eliminating the erroneous positive predictions outside the object to a large extent. Since the filtering step should eliminate noise it is expected that the filtering will also have a positive effect on segmentation performance. Results of this experiment are presented and discussed in the next chapter.

6.3 Center Location

The goal of the second experiment is to show the ability of the automatic Vickers indentation inspection algorithm to determine the center of the indentation. Determining the center of the indentation is an important step in subsequent high-level feature extraction and processing operations. The final image segmentation, produced and evaluated in the first experiment, is taken as a starting point. The center position is determined by averaging the positions of pixels that form the indentation in the segmented image. In other words, the position of those pixels that were predicted as part of the indentation were averaged to obtain the center location.

The indentation's center as determined by the algorithm was compared with the ability of the human experts. Five human experts marked the center of the indentation in every image of the sequence. This process was repeated two more times. Therefore, we obtained 15 attempts (5 experts, each with 3 attempts for every image) for every image in the sequence. These attempts were averaged to obtain an average human attempt for every image. Both unfiltered and filtered predictions were used for the calculation of the center location and compared with the results of the average human expert. The results obtained using the filtered predictions were expected to exhibit lower dispersion throughout the whole tested image sequence.

For training of an SVM model, 12 images were selected, so as to form a sufficiently representative sample of the whole sequence of 42 images. Figure 6.3 shows a diagram of an image sequence beginning with the 11th image. This is because the first 10 images had to be thrown way due to overexposure. The black dots mark the images used for training and the white dots images used for testing. As the whole image sequence is too long, the presence of the testing images in the middle of the sequence was indicated by the small dots to conserve space.

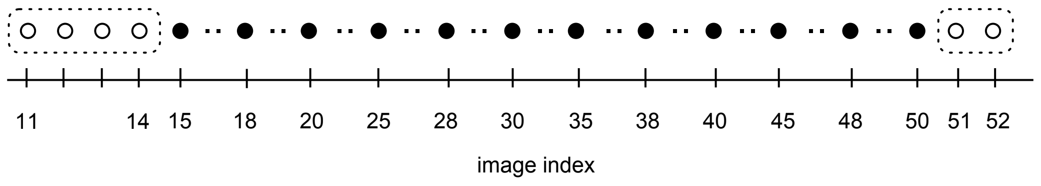


Figure 6.3: Diagram illustrating the choice of training images in the context of the whole image sequence. Training images (black); testing images (white and small black dots).

If the SVM is provided in the training phase with the training images that bound a given subsequence of testing images, the SVM should generalize well to the unseen intermediate testing images (i.e. achieve reasonable segmentation performance). Thus, it is assumed that the quality of image segmentation will be better for the testing images lying in between the training images (black dots, Fig. 6.3) rather than for those lying on either end of the sequence (marked by dashed boxes, Fig.6.3).

Since all of the labeled images were used for training of the classifier, it is impossible to quantify the performance of the classifier in this experiment. The learned SVM model is only used to provide predictions and it is only through visual inspection that one is able to roughly assess the quality of image segmentation.

Chapter 7

Experiment Results and Discussion

In this chapter, results of the both experiments explained in the previous chapter are presented and discussed.

7.1 Setting of the SVM

Before experiment results can be obtained, a suitable implementation of an SVM needs to be chosen and properly configured.

When deciding to use an SVM for solving a specific task, one does not need to write their own implementation from scratch. SVMs have become very popular over the years and many different implementations of the SVM classifier are available to the public. For the early experimentation and optimal parameter setting, the LibSVM implementation [8] was used. LibSVM has a large user base and many projects are built on it. These include wrappers for various programming languages such as Python, R, C#, C++ and Matlab, as well as parallel implementations utilizing CUDA and MPI. It provides support for solving classification problems (C-SVC, ν-SVC) and regression problems (ϵ-SVR, ν-SVR). The most popular kernel choices mentioned in Section 4.4, such as linear, polynomial, RBF and sigmoid kernel, are all supported as well.

For the final experiments, where larger data sets were handled, LibSVM implementation was still very inefficient, as it utilized only one CPU core, and training was taking too long (e.g. training on 2 images took 31 minutes; training on 4 images 109 minutes). For this reason, a decision was made to use πSVM [6], a parallel implementation of the SVM classifier, which can fully exploit the power of the six-core processor at our disposal. The great advantage of πSVM is that it builds on top of LibSVM, and thus has largely the same user interface. The speed-up achieved with this parallel implementation was very noticeable. To compare, training on 4 images (230,400 feature vectors) using LibSVM took 109 minutes, while training on 6 images (345,600 feature vectors) with πSVM utilizing all six cores took 99 minutes.

Chapter 4 of this thesis mentions the fact that a kernel function needs to be chosen in order to use an SVM classifier for nonlinear classification. Unfortunately, the choice of the kernel function is something of a black art, which is to say that no method for choosing the appropriate kernel exists and the choice is left to the intuition of the designer. The RBF kernel was chosen for the purposes of this thesis, as it is a popular choice in many applications and because it provided the best results of all tested kernels.

Other things that need to be considered when using an SVM classifier are the optimal choices of the values of the regularization parameter C and kernel parameter γ (for RBF kernel). As it turns out, finding an optimal setting of these parameters is of particular importance and should not be underestimated, as it can very significantly affect the final classification performance. To find the optimal parameter settings of the SVM classifier, a grid search method was used. A grid search takes into account all the configurations of the classifier parameters (C, γ). For every such configuration, one SVM model is trained and classification performance is evaluated. The parameter configuration with the highest classification performance is, naturally, the optimal one. The values of the regularization parameter C should form an exponentially growing sequence such as $C = (2^{-5}, 2^{-3}, \ldots, 2^{-15})$ and for the kernel parameter the sequence $\gamma = (2^{-15}, 2^{-13}, \ldots, 2^{3})$ is suggested [18]. Nevertheless, as the available time for experiments and computational resources were limited, utilizing this method for parameter selection would have been prohibitively computationally expensive. For this reason, optimal parameter values were determined by experimentation. While parameter γ was fixed, the value of parameter C was gradually increased only if the change contributed to the classification performance. In this way $C = 1000$ was chosen. The same processes was repeated for kernel parameter, this time with regularization parameter C fixed. The value of the kernel parameter was finally set to $\gamma = 1024$. As the various parameter configurations were tested, it is worth noting that not only the classification performance was affected by the varying parameters, but also the time required for training of the classifier.

Nonetheless, even after setting the optimal parameters (C, γ), training of the SVM classifier was still taking too long even for smaller datasets. As described in Chapter 4, training of a support vector machine involves finding a solution to the convex optimization problem. Both LibSVM and πSVM employ the Sequential Minimal Optimization algorithm, which is an iterative optimization technique and therefore requires some stopping condition to be provided. To decrease the training time, while preserving the classification performance, an optimal stopping condition of $e = 2$ was found to be the best. This choice of stopping condition significantly reduced the training time, while the drop suffered in classification performance was negligible (a tenth of a percent) compared to using the default stopping condition ($e = 0.001$).

7.2 Segmentation Performance Results

The performance of the SVM classifier was evaluated for the original unfiltered predictions (Table 7.1) as well as the filtered predictions (Table 7.2). The models were trained using the parallel πSVM implementation utilizing six CPU cores. The process of training the SVM lasted from 15 minutes to 10 hours depending on the training set size. The filtering of predictions proved to be beneficial. Figure 7.1 compares predictions obtained for the first testing image. As seen in the figure, the presence of holes and noise is eliminated.

As expected, the values of all the performance measures are at their lowest for the smallest training set sizes. Looking at the results in Table 7.1, it is apparent that increasing the training set size from two images to four images yields approximately a 3% increase in precision and a 10% increase in recall. However, a further increase in training set size contributes only to the

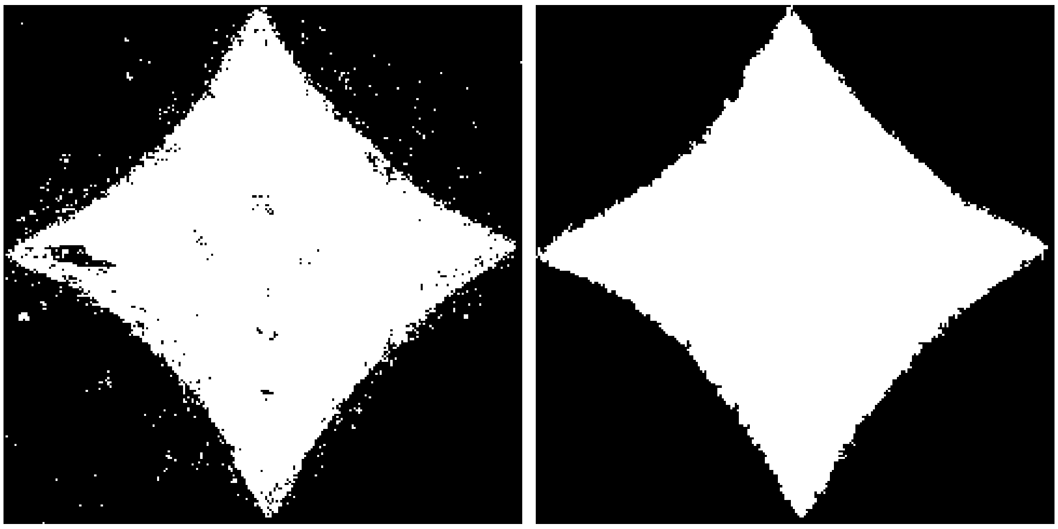


Figure 7.1: Comparison of the original unfiltered predictions (left) and predictions after the filtering (right).

# of train. images	Precision	Recall	F_1-score	Matthews
2	0.939	0.835	0.884	0.837
4	0.962	0.934	0.948	0.920
6	0.965	0.949	0.957	0.934
8	0.960	0.963	0.962	0.941
10	0.964	0.969	0.966	0.949

Table 7.1: Segmentation performance evaluation for unfiltered predictions.

classifier's recall, F_1-score and MCC. Precision levels off at approximately 96%. In Table 7.2, we observe a decrease in precision despite an increase in training set size. Nevertheless, the F_1-score still goes up, because with increasing training set size recall increases.

Comparison of values in Tables 7.1 and 7.2 reveals an overall increase in all the performance measures. The F_1-score starts at 90.4%, jumps to 95.4% and then slowly reaches 97.4% for the largest training dataset. The Matthews coefficient behaves in a similar manner. It is clear that the filtering of predictions definitely has a positive effect on the segmentation quality, which comes as no surprise.

# of train. images	Precision	Recall	F_1-score	Matthews
2	0.983	0.837	0.904	0.917
4	0.985	0.933	0.958	0.960
6	0.979	0.948	0.963	0.963
8	0.975	0.962	0.968	0.967
10	0.978	0.969	0.974	0.969

Table 7.2: Segmentation performance evaluation for filtered predictions.

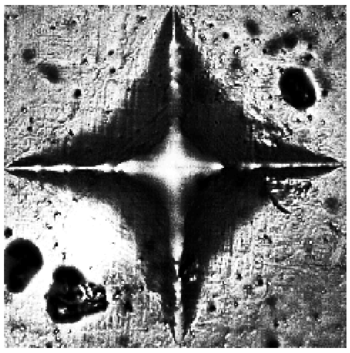
7.3 Center Location Results

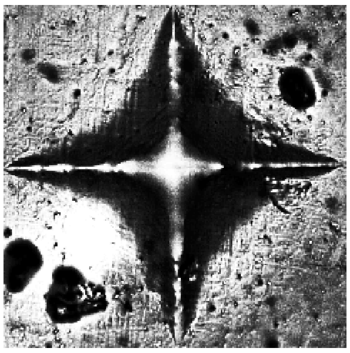
Undoubtedly, obtaining results of this experiment was the most computationally demanding task of all the experiments. Making use of all the six cores of the AMD FX-6300 CPU, the training of the SVM model on the twelve labeled images lasted approximately 12 hours.

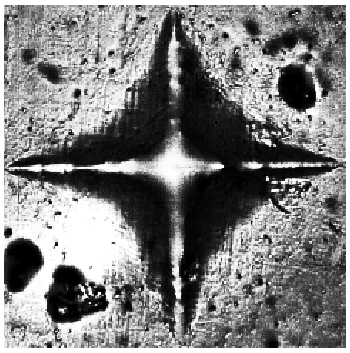
At the end of the image sequence, there is a noticeable sudden abrupt change in the polymer indentation recovery. Figure 7.2 shows how the segmentation quality decreases when the images taken from the end of the sequence are used for testing. The last image that the classifier considered was image 50. However, the SVM was still able to generalize further down the sequence, though not so well anymore, because no image beyond the image 50 was used for training. That is why the prediction 51 in fig. 7.2 is noticeably worse than the prediction 49. The filtering step is again very helpful in removing noise.

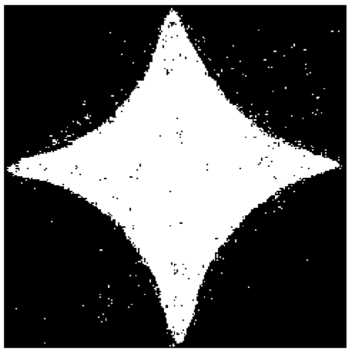
The graph in Figure 7.3 shows how the X and Y coordinates of the indentation's center develop. The impact of using the filtered predictions can be seen at the beginning and the end of the sequence.

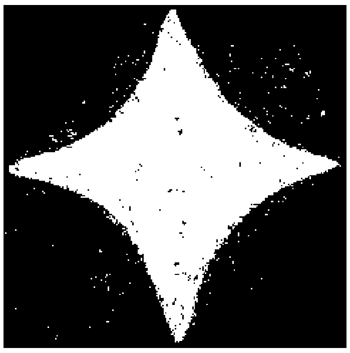
With the assumption in mind, that the position of the true center changes minimally or does not change at all from image to image, all the averaged human attempts for every image were again averaged (across images in the sequence). This way, a single number was obtained for the whole image sequence, which represents the mean human center position for all images in the sequence. In the same way, the average position as determined by the computer was calculated. To see the impact of filtering, both unfiltered and filtered predictions were considered. Table 7.3 sums up the mean center

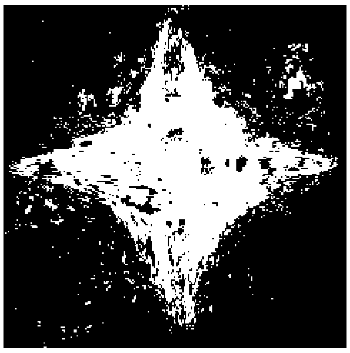

(a) image 47

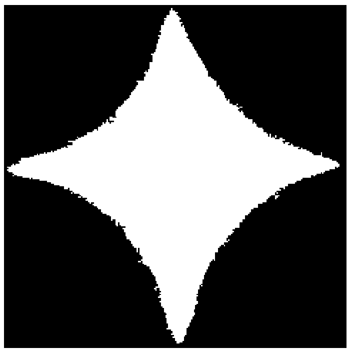

(b) image 49

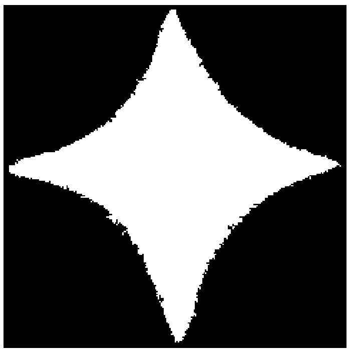

(c) image 51


(d) prediction 47


(e) prediction 49


(f) prediction 51


(g) filtered pred. 47


(h) filtered pred. 49

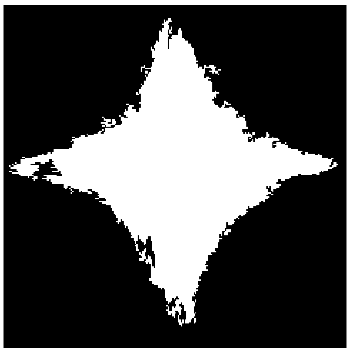

(i) filtered pred. 51

Figure 7.2: First row: input testing images indexed 47, 49 and 51. Second row: corresponding predictions as generated by the SVM. Third row: filtered predictions. More rapid change in indentation recovery is noticeable in images 49 and 51.

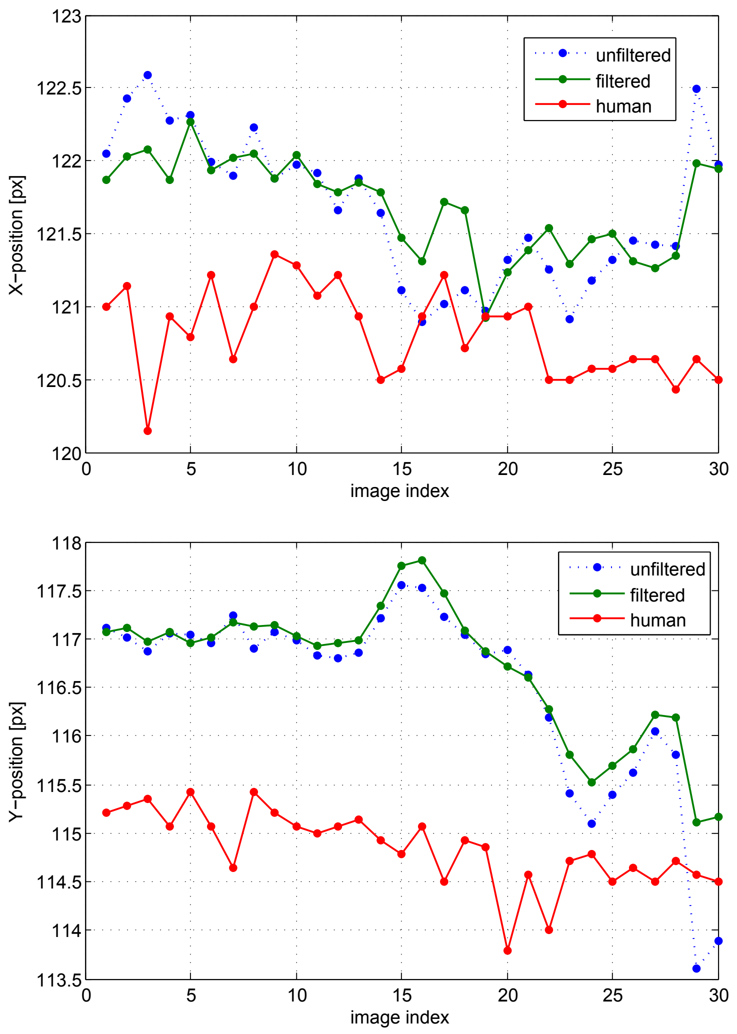


Figure 7.3: X and Y-coordinate development of the center location as determined by human and computer for individual images in the sequence.

position in pixels and its variance for individual coordinates.

The human attempts have the highest variance in both coordinates. This is attributed to the fact that the attempts were made by five human experts, each having their own perception of the center location. The hypothesis that using the filtered predictions for center determination will yield more precise estimates of the center location was confirmed.

To further compare the results obtained by the human and the computer, the mean human center position was taken as the reference point to which

	mean center position [px]		variance [px]	
	X	Y	X	Y
human	120.82	114.85	1.3485	1.9529
unfiltered	121.67	116.49	0.2546	0.9662
filtered	121.69	116.70	0.1078	0.4999

Table 7.3: Comparison of mean center position and variance as determined by human and computer.

all computer attempts at locating the center were compared. The difference between the mean human center position and the center estimate provided by the algorithm for every image in the sequence was calculated. Table 7.4 lists the mean and variance of the error along with the total error.

	unfiltered		filtered	
	X	Y	X	Y
mean absolute error	0.6831	4.6525	0.6934	4.4418
absolute error variance	0.2344	0.9662	0.1001	0.4999
sum of absolute error	20.494	139.5763	20.8009	133.2534

Table 7.4: Mean absolute error, absolute error variance and total absolute error in center location as compared to mean human center location.

Table 7.4 reveals an interesting phenomenon. The mean absolute error is always larger for Y-axis. One can only speculate as to why this phenomenon occurs. Clearly, as the center position is calculated by averaging, the predictions along the vertical axis were skewed in one direction or another. This could be caused by the fact that the cross-like ridge left by the indenter (visible in the input images) is more pronounced in the lower corner of the indentation then it is in the upper corner, thus possibly causing the biased position estimates.

Chapter 8

Conclusion

The main goal of this thesis was the design of suitable features and implementation of an algorithm for segmentation of Vickers indentation images. Segmentation of Vickers indentations is necessary for tracking of indentation boundaries, which in turn serves for non-destructive polymer quality evaluation.

Chapter 2 provides basic information about the Vickers hardness test. A brief overview of image segmentation techniques can be found in chapter 3. A support vector machine classifier (Chapter 4) was chosen, because of its robustness, as a means for solving the image segmentation problem. This required the creation of a training dataset, which included provision of ground-truth segmentation (labeling) by the human expert.

The design of suitable features and feature vector formation was another necessary step. The variance features introduced in Chapter 5 have a very good discriminatory value and proved to be suitable for the problem at hand, as evidenced by the high segmentation performance summarized in Tables 7.1 and 7.2. Precision, recall, F_1-score and Matthews Correlation Coefficient were measures employed for the evaluation of the image segmentation process. Because the image segmentation was performed per-pixel, the SVM predictions obtained were slightly noisy. To deal with this problem, filtering of the predictions was necessary.

The first experiment was designed to measure the quality of image segmentation as well as the learning ability of the SVM classifier. The highest segmentation quality was achieved for the largest training dataset (10 images), where the F_1-score reached 97.4%. However, the segmentation quality can be considered very good even for the smallest training set size (2 images), especially when filtering of predictions is employed.

The second experiment concentrated on comparing the ability of human experts and the designed algorithm to determine the center of the indentation. The results summarized in Tables 7.3 and 7.4 indicate that the center positions determined by the algorithm have lower variance than the human-located center positions in both vertical and horizontal axes. It is also apparent that the variance in the Y-axis is always larger than the variance in the X-axis. Furthermore, it is evident that using the filtered predictions lowers the center position variance in both axes.

To conclude, microscopic images of indentations exhibit a lot of variation in brightness, texture and lighting conditions. If the algorithm proposed in this thesis is to be used in industry, further testing of the proposed features is necessary on more extensive datasets.

Bibliography

[1] M. A. Aizerman, E. A. Braverman, and L. Rozonoer. “Theoretical foundations of the potential function method in pattern recognition learning.” In: *Automation and Remote Control,* Automation and Remote Control, 25. 1964, pp. 821–837.

[2] D.R. Askeland, P.P. Fulay, and D.K. Bhattacharya. *Essentials of Materials Science and Engineering: SI Edition.* Essentials of materials science and engineering. Cengage Learning, 2009.

[3] D. H. Ballard and C. M. Brown. *Computer Vision.* 1st. Prentice Hall Professional Technical Reference, 1982.

[4] D. Barber. *Bayesian Reasoning and Machine Learning.* Cambridge University Press, 2012.

[5] C. Bishop. *Patter Recogniton and Machine Learning.* Springer, 2006.

[6] D. Brugger. “Parallel Support Vector Machines”. In: *Proceedings of the IFIP International Conference on Very Large Scale Integration of System on Chip.* Springer, Oct. 2007.

[7] W. D. Calister and D. G. Rethwisch. *Fundamentals of Materials Science and Engineering: An Integrated Approach.* 3rd. Wiley, 2008.

[8] C.-C. Chang and C.-J. Lin. “LIBSVM: A library for support vector machines”. In: *ACM Transactions on Intelligent Systems and Technology* 2 (3 2011). Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, 27:1–27:27.

[9] C. Cortes and V. Vapnik. “Support-vector Networks”. In: *Machine learning* 20.3 (1995), pp. 273–297.

[10] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods.* Cambridge University Press, 2000.

[11] E.R. Davies. *Machine Vision: Theory, Algorithms, Practicalities.* 3rd. Morgan Kaufmann Publishers Inc., 2005.

[12] J. Friedman, T. Hastie, and R. Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Vol. 1. Springer Series in Statistics, 2009.

[13] M. Gadermayr, A. Maier, and A. Uhl. "Active contours methods with respect to Vickers indentations". In: *Machine Vision and Applications* (2012), pp. 1–14.

[14] M. Gadermayr, A. Maier, and A. Uhl. "The AreaMap Operator and its Application to Vickers Hardness Testing Images". In: *International Journal of Future Generation Communication and Networking* (2012).

[15] R. M. Haralick and L. G. Shapiro. "Image segmentation techniques". In: *Computer Vision, Graphics, and Image Processing* 29.1 (Jan. 1985), pp. 100–132.

[16] S. L. Horowitz and T. Pavlidis. "Picture Segmentation by a Tree Traversal Algorithm". In: *Journal of the ACM* 23.2 (Apr. 1976), pp. 368–388.

[17] M. Hrúz, J. Široký, and D. Maňas. "Particle Swarm Optimization for Automatic Hardness Measurement". In: *Chemické listy* (106 2012), pp. 434–437.

[18] C.-W. Hsu, C.-C. Chang, and C.-J. Lin. "A Practical Guide to Support Vector Classification". In: (2010). http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

[19] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988.

[20] K.P. Murphy. *Machine Learning: A Probabilistic Perspective.* MIT Press, 2012.

[21] N. Otsu. “A threshold selection method from gray-level histograms”. In: *IEEE Transactions on Systems, Man and Cybernetics* 9.1 (Jan. 1979), pp. 62–66.

[22] B. Schölkopf and A.J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond.* MIT press, 2001.

[23] W. E. Snyder and H. Qi. *Machine Vision.* Cambridge University Press, 2010.

[24] M. Sonka, V. Hlaváč, and R. Boyle. *Image Processing, Analysis, and Machine Vision.* Thomson-Engineering, 2007.

[25] R. Szeliski. *Computer Vision: Algorithms and Applications.* 1st. Springer-Verlag New York, Inc., 2010.

[26] C. A. Tweedie and K. J. Van Vliet. “On the Indentation Recovery and Fleeting Hardness of Polymers”. In: *Journal of Materials Research* (2006).

[27] G.F. Vander Voort and R. Fowler. “Low-load Vickers Microindentation”. In: *Advanced Materials & Processes* 170 (4 Apr. 2012), pp. 28–33.

[28] V. Vapnik. *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics.* Springer-Verlag New York, Inc., 1982.

[29] V. Vapnik. *Statistical Learning Theory.* Wiley, 1998.

[30] J. Yu and X. Aiwei. “A New Method for Automatically Measurement of Vickers Hardness Using Thick Line Hough Transform and Least Square Method”. In: *Image and Signal Processing, 2009. CISP ’09. 2nd International Congress on.* 2009, pp. 1–4.

[31] Q. Zhou, G.-Z. Yan, and Y. Zhang. “A New Method for Quick and Automatic Analysis of the Image of Vickers Hardness Using Wavelet Theory”. In: *Acta Metrologica Sinica* (2005).