Developing Robust Algorithms for Feature Extraction in Images of Polymer Layers

Anton Wemmenborn

Master's thesis
2019:E7

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematical Statistics

LUND UNIVERSITY

Abstract

Faculty of Engineering LTH
Centre for Mathematical Sciences

Master of Science

Developing robust algorithms for feature extraction in images of polymer layers

by Anton Wemmenborn

Automated manufacturing processes are an important component of today's industries. Assuming the processes are properly maintained they allow for great efficiency when producing various goods. Image analysis can be used as a tool to monitor such processes and evaluate their results.

This Thesis treats development of algorithms for automatic evaluation of images acquired using a standardized procedure. The images contain polymer samples from which the relative positions of two lines are extracted. These positions are thought to be related to machine settings. The image analysis algorithms have to be robust to large variation among the acquired images. Lighting and colour in general as well as shape, location and colour of the two sought lines will vary between samples. Recurring features of the images can be used as a basis of evaluation.

The algorithms use several different techniques in conjunction to identify the lines. Pyramid reduction is used to reduce noise in the images and computational effort. Principal component analysis is used to reduce the number of dimensions treated from three to two. The polymer sample and one of the lines are found using thresholding and morphological operations with parameters automatically calculated using certain parts of the images. The second line is found as the path of most likely pixels considering their intensity, gradient magnitude, gradient direction and location. Finally the relative positions of the lines are examined using principal component analysis.

The resulting algorithm produces relatively good results but has room for improvement, and suggestions for further work are provided. Machine settings are found to influence but not fully explain the relative positions of the two lines.

Acknowledgements

I would like to thank my supervisors Eskil Andreasson and Daniel Zulumovski at Tetra Pak® for their guidance and advice as well as putting up with my countless questions about polymer samples.

Further I would like to thank my supervisor Johan Lindström at the Centre for Mathematical Sciences for his feedback and constructive advice providing invaluable guidance for problems encountered.

Finally I would like to thank my family and friends for their continuous support during my period of studies. In particular I would like to thank my wife and daughter for their encouragement as well as enduring my rants about image analysis.

Contents

List of Symbols

I	Image
I_{in}	Input image
I_{out}	Output image
M	Amount of rows in image matrix
N	Amount of columns in image matrix
$p(x,y)$	Pixel at column x and row y
$h(x,y)$	Structuring element centered at column x and row y
$N_4(p(x,y))$	4-Neighbourhood around $p(x,y)$
$N_8(p(x,y))$	8-Neighbourhood around $p(x,y)$
T	Threshold
PC_i	Principal component i
γ_i	Fraction of total variance explained by PC_i
L	Line
μ	Expected value
σ	Standard deviation

1 Introduction

1.1 Background

Automated manufacturing processes are an important component of today's industries. Assuming the processes are properly maintained they allow for great efficiency when producing various goods. This Thesis will treat image analysis, an important tool for maintaining and optimizing automated processes. Image analysis can be used as a tool to visually monitor an ongoing process and in case any serious deviations occur a warning could be issued or an automatic response triggered. Image analysis can also be used to optimize certain automated processes by evaluating the results and making improvements where possible.

This report is written at and in collaboration with Tetra Pak®, for which automated processes are of great interest. To add to the various tools already used when monitoring and analyzing automated processes at Tetra Pak® this Thesis is aimed at developing robust algorithms for feature extraction in images of polymer.

1.2 Thesis objective

This Thesis seeks to produce robust algorithms to analyze input images of polymer samples acquired using a standardized technique at Tetra Pak®. In particular it seeks to find two lines covering the entire width of the samples and examine their relative positions for different machine settings.

2 Data

2.1 Data acquisition

The algorithms developed in this Thesis will deal only with input images of polymer samples acquired in a process standardized at Tetra Pak®. First the polymer samples are preprocessed and an interesting region of the polymer in every sample is manually colored by red ink to increase the contrast to its neighbouring regions. Several images of a single polymer sample are then taken in a custom built camera rig, after which these images are stitched together to one high resolution image of the sample. The algorithms developed in this Thesis will deal with those final high resolution images.

The images acquired are separated into a modelling data set with 30 samples used during development of the algorithms, a validation data set with 30 samples to be closely examined and finally an extended validation data set with 146 samples used to evaluate how machine settings impact the sought lines.

2.2 Delimitations and assumptions

Due to the standardized image acquisition process several assumptions can be made about the data. The input images are composed in a very similar manner. The polymer samples examined are positioned horizontally at the center of the image with plenty of pixels to its left, right and bottom side. The samples are illuminated by a light source positioned horizontally in the background, leaving the top left and the top right corner of the input image unilluminated.

There will be two continuous lines of great interest apparent in every input image. One is positioned directly below the region coloured by red ink and will therefore be referred to as the colour line. The second is positioned in a polymer region and is detected as a dark continuous shadow. The second line will be referred to as the shadow line. The two sought lines cover the entire width of a sample.

Further the acquired images are produced at three different machine settings referred to as low, medium and high settings. These will not impact how the lines are extracted but might affect the relative positions of the lines.

Only small parts of input images sufficient to explain the algorithms will be presented in this report since Tetra Pak® does not wish to disclose more detail than necessary regarding the image acquisition process or the composition of the input images. The first example can be seen in figure 2.1 where the colour line and shadow line close to the left border of a sample can be seen.

The lighting of the samples will be very similar between images since they are mostly illuminated by a bright light source in the background. Small variations caused by ambient light are bound to occur but the brightness of the background light source will ensure that those variations are small.

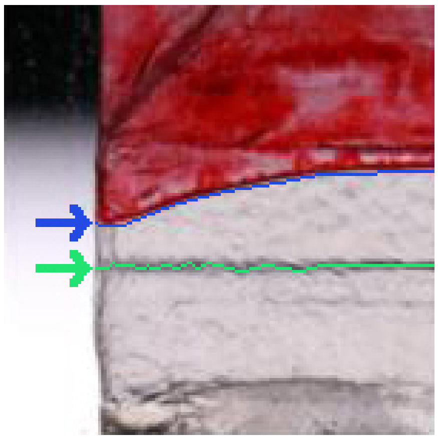


FIGURE 2.1: The left border of a sample. The colour line is manually marked in blue and the shadow line is manually marked in green.

Further the colour of the polymer will not vary between the images since the same material is used.

The intensity of the red ink in the manually coloured region has large variance. This is because of the manual application of the ink and its natural tendencies to accumulate in small nooks and cavities.

3 Theory

This chapter describes the foundations upon which the algorithms developed are built. It starts with a brief description of digital images. Some essential parts of image analysis are then described. The interested reader can get a more thorough understanding of image analysis in (Sundararajan, 2017; Peters, 2017). Finally a short section about principal component analysis follows, a tool of great importance to the developed algorithms.

3.1 Digital image representations

Digital images contain discrete two dimensional representations of three dimensional objects. A single digital image I is stored digitally as an array with dimensions $M \times N \times C$, where M denotes the number of rows, N denotes the number of columns and C denotes the number of colour channels in the image. Every point in the array is called a pixel and the value $p(x, y, z)$ describes the intensity at column x, row y and colour channel z. If the image has only one colour channel the notation is often simplified to $p(x, y)$. The intensity at a given location is proportional to the brightness of the scene at that point in the image (Sundararajan, 2017). A reader not used to the coordinate system of an image should keep in mind that it commonly starts in the top left corner. This Thesis will deal with three types of digital image representations, RGB images, grayscale images and binary images.

3.1.1 RGB images

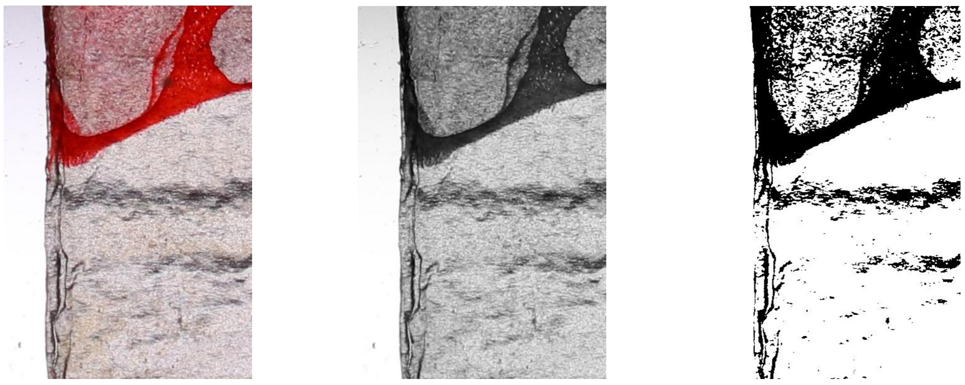
RGB stands for Red, Green and Blue and describes the three colour channels included in the image representation. Every pixel in an RGB image is described by an 1×3 array whose entries are the red, green and blue colour intensities (Peters, 2017). An example of an RGB image can be seen in figure 3.1a.

3.1.2 Grayscale images

Grayscale images are represented as two dimensional matrices of size $M \times N$. Only one colour channel is considered and every entry in the matrix corresponds to a colour on the black to white spectrum. Bright pixels have high intensities and dark pixels have low intensities (Peters, 2017). An example of a grayscale image can be seen in figure 3.1b.

3.1.3 Binary images

Finally we have binary images. An example of such an image can be seen in figure 3.1c. A binary image is two dimensional with size $M \times N$ since it only contains a single colour channel. In fact it can only take two intensity values in its single colour channel. Every pixel is assigned either the colour black with intensity zero or the colour white with intensity one (Peters, 2017). Binary images are commonly obtained by thresholding grayscale images.



(a) RGB representation. (b) Grayscale representation. (c) Binary representation.

FIGURE 3.1: The three different image representations illustrated. From left to right an RGB, grayscale and binary image representation of the same object can be seen.

3.2 Image analysis

3.2.1 Pixel neighbourhoods

The neighbourhood of a pixel is defined as a set of pixels close to it. Given a pixel p with coordinates (x, y) in an image, the $4-$neighbourhood is defined as the pixel itself and the pixels directly adjacent to it vertically and horizontally, see figure 3.2a. Mathematically it is defined as (Peters, 2017; Sundararajan, 2017):

$$N_4(p) = \{p(x,y), p(x-1,y), p(x+1,y), p(x,y-1), p(x,y+1)\} \tag{3.1}$$

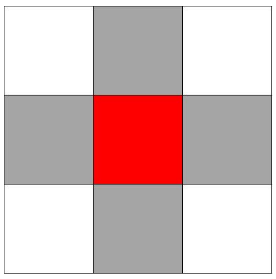
The $8-$neighbourhood is the union of the $4-$neighbourhood and the pixels diagonally adjacent to p, see figure 3.2b, and is defined as (Peters, 2017; Sundararajan, 2017):

$$\begin{aligned} N_8(p) = \{&p(x-1,y-1), p(x,y-1), p(x+1,y-1), p(x-1,y), p(x,y), \dots \\ &p(x+1,y), p(x-1,y+1), p(x,y+1), p(x+1,y+1)\}. \end{aligned} \tag{3.2}$$

3.2.2 Image gradients

In image analysis an important operation is to detect edges in an image. These edges are characterized as pixels with a sudden variation in intensity compared to their neighbouring pixels, and often mark interesting features of the image (Szeliski, 2011).

A common way to detect edges is to examine the image gradients at every point in the image, since the gradient should be large where there are large sudden variations in intensity. The gradient in the x direction of an image is given by $g_x(x,y) = p(x+1,y) - p(x-1,y)$, and the gradient in the y direction is defined accordingly (Sundararajan, 2017).


(a) The 4−neighbourhood.

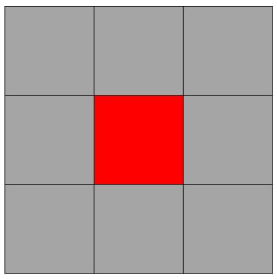

(b) The 8−neighbourhood.

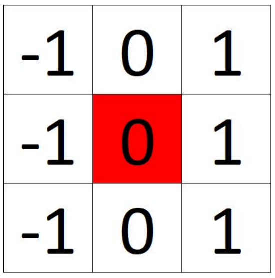
FIGURE 3.2: The 4− and 8−neighbourhoods. The center pixels are marked in red and the other pixels belonging to the neighbourhoods are marked in gray.

To deal with the entirety of an image at once it is useful to apply gradient operators or filters. A common choice is the Prewitt gradient operator mask. One operator mask in either direction is applied to the 8−neighbourhood of a pixel, and the sum of the elements of those operator masks is assigned to be the gradient of the centre pixel in the corresponding direction. Using operator masks reduces the influence of noise in the image by averaging over pixels in the neighbourhood (Sundararajan, 2017).

The Prewitt gradient operator calculates the gradient in the x direction by the following formula, corresponding to application of the gradient mask in figure 3.3 (Sundararajan, 2017).

$$\begin{aligned} g_x(x,y) = &\big(p(x+1,y-1) + p(x+1,y) + p(x+1,y+1)\big) \\ &-\big(p(x-1,y-1) + p(x-1,y) + p(x-1,y+1)\big). \end{aligned} \tag{3.3}$$

The gradient in the y direction, g_y, is calculated by simply transposing the operator mask in the x direction (Sundararajan, 2017).



-1	0	1
-1	0	1
-1	0	1

FIGURE 3.3: The Prewitt operator mask in the x direction. The result of applying the operator mask to the 8−neighbourhood of a pixel is assigned to the central pixel marked in red.

The direction of the gradient with respect to the row axis as well as the magnitude of the gradient can then be calculated according to (Sundararajan, 2017):

$$g(x,y) = \sqrt{\left(g_x^2(x,y) + (g_y^2(x,y)\right)}, \tag{3.4}$$

$$\theta(x,y) = tan^{-1}\left(\frac{g_y(x,y)}{g_x(x,y)}\right), \tag{3.5}$$

where $g(x,y)$ denotes the gradient magnitude and $\theta(x,y)$ denotes the gradient direction.

3.2.3 Thresholding

Thresholding is a basic segmentation operation working at pixel level. During thresholding the value assigned to every pixel is compared to a threshold or set of thresholds. These thresholds split all possible values a pixel can hold into intervals, and during the operation every pixel is assigned a new value based on which interval the original value falls into (Sundararajan, 2017; Jähne, 2005).

The basic case of thresholding applies a single threshold to an image. This yields two different intensity intervals, one below the threshold and one above, resulting in a binary image (Sundararajan, 2017; Jähne, 2005). The basic process of applying a threshold to an image can be seen in figure 3.4 where a grayscale image is thresholded to a binary image.

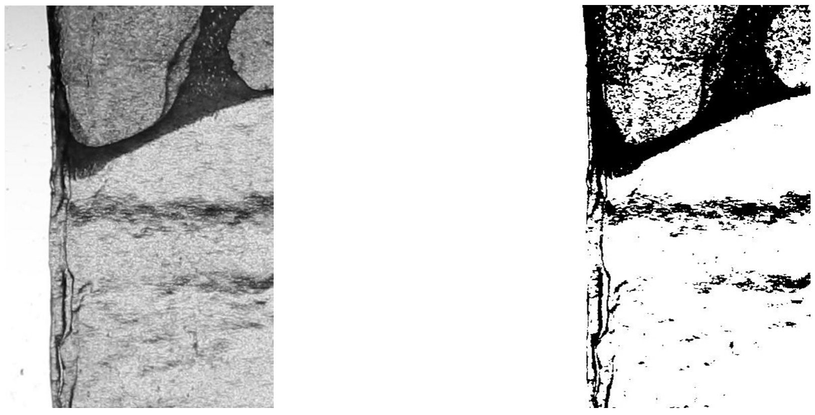

(a) Grayscale image. (b) Binary image.

FIGURE 3.4: The first subfigure is thresholded to split the intensities into two equally large intervals. The resulting binary image can be seen in the second subfigure.

3.2.4 Morphological operations

Morphological operations are carried out on binary images, adding or removing pixels. The two basic morphological operations are dilation and erosion, which respectively add and remove pixels at the border of objects in the image (Sundararajan, 2017; Jähne, 2005).

Dilation is an operation which fills in small holes and makes the borders of objects in the image smoother. For dilation a binary structuring element $h(x, y)$ is chosen. If we denote the input image $I_{in}(x, y)$ and the output image $I_{out}(x, y)$ dilation is similar to a linear convolution of logical operations. For every pixel in the input image the neighbourhood defined by ones in the structuring element centered at that pixel is considered, and if at any point a one in the structuring element $h(x, y)$ and the input image $I_{in}(x, y)$ coincide, the dilation will yield $I_{out}(x, y) = 1$. Mathematically we have the following for a single pixel if we use the four neighbourhood as a structuring element (Sundararajan, 2017; Jähne, 2005).

$$h(x, y) = N_4 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{3.6}$$

$$I_{out}(x, y) = I_{in}(x-1, y) \mid I_{in}(x, y-1) \mid I_{in}(x, y) \mid I_{in}(x, y+1) \mid I_{in}(x+1, y), \tag{3.7}$$

where | denotes the OR operator, which means that if either of the included pixels in I_{in} has the value one, then $I_{out}(x,y) = 1$. An example of dilation can be seen in figure 3.5.

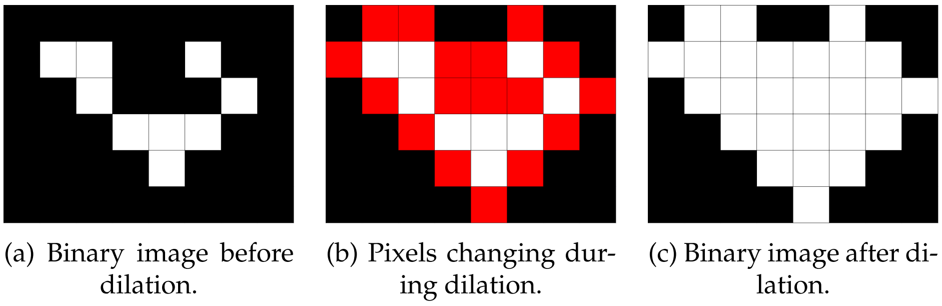

(a) Binary image before dilation. (b) Pixels changing during dilation. (c) Binary image after dilation.

FIGURE 3.5: The dilation operation using the N_4 structuring element. The first subfigure shows I_{in}, the second subfigure shows red pixels changing value as a result of the operation and the third subfigure shows I_{out}.

A similar operation of equal importance is erosion. As the name of the operation makes evident it shrinks objects in the image. Small objects may disappear and objects may become disconnected into several smaller objects. The operation is similar to dilation, but using the logical AND operator, &, instead of the OR operator, |. This means that for a pixel of the output image to take the value one, $I_{out}(x,y) = 1$, every point of the input image corresponding to a one in the structuring element must also have the value one. An example of erosion can be seen in figure 3.6. Mathematically we have the following if we use the four neighbourhood as structuring element (Sundararajan, 2017; Jähne, 2005).

$$h(x,y) = N_4 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} . \tag{3.8}$$

$$I_{out}(x,y) = I_{in}(x-1,y) \,\&\, I_{in}(x,y-1) \,\&\, I_{in}(x,y) \,\&\, I_{in}(x,y+1) \,\&\, I_{in}(x+1,y). \tag{3.9}$$

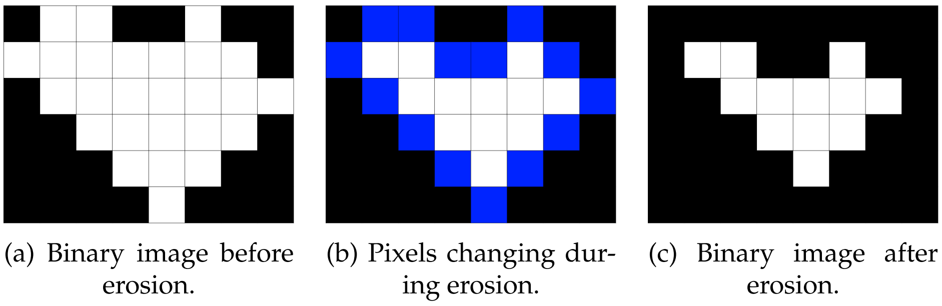

(a) Binary image before erosion. (b) Pixels changing during erosion. (c) Binary image after erosion.

FIGURE 3.6: The erosion operation using the N_4 structuring element. The first subfigure shows I_{in}, the second subfigure shows blue pixels changing value as a result of the operation and the third subfigure shows I_{out}.

We now consider the morphological operations opening and closing, which are both very useful and very simple. Opening consists of choosing a single structuring element $h(x,y)$ and applying erosion followed by dilation, effectively removing small objects in the binary image. The operation will also affect the contour of objects by removing narrow portions, potentially splitting larger objects into several smaller ones (Sundararajan, 2017; Jähne, 2005).

Closing, on the other hand, applies dilation first and erosion second. This does not generally expand the objects of the image but it can merge objects together when they are only separated by a narrow region of zeros (Sundararajan, 2017; Jähne, 2005). This effect can be seen when looking at figures 3.5a and 3.6c. Taking the image in figure 3.5a as input and performing the closing operation with a $4-$ neighbourhood yields the image in figure 3.6c.

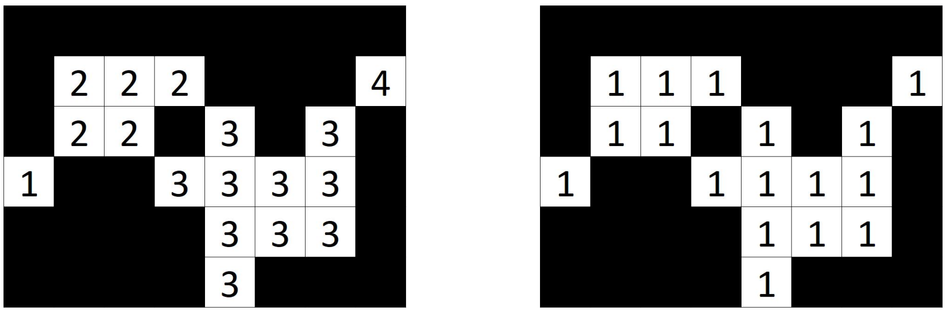
3.2.5 Connected components

A connected component in a binary image is defined as the set of ones where every object in the set is connected to at least one other object in the set by a chosen structuring element $h(x, y)$. Usually the connected components of an image are defined using either the four or the eight neighbourhoods, N_4 or N_8. Sets of connected components can be found using a flood fill algorithm which iteratively adds eligible pixels to the set (Sundararajan, 2017; Szeliski, 2011).

The flood fill algorithm starts with a single pixel in the image as the first object in the connected set. A set image, I_1, of equal size to the input image, I_{in}, is then created. Every pixel which is not in the set is given the value zero. The algorithm then iteratively calculates new set images I_i until $I_i = I_{i-1}$ according to (Sundararajan, 2017)

$$I_i = \left(I_{i-1} \oplus h\right) \,\&\, I_{in}, \tag{3.10}$$

where $\oplus$ is the dilation operation and I_{in} is the input image. A comparison of all connected components found in a small binary image using different structuring elements can be seen in figure 3.7. Note that the image has four sets of connected components if N_4 is chosen as structuring element but only one set of connected components when N_8 is chosen.



(a) N_4 structuring element. (b) N_8 structuring element.

FIGURE 3.7: Comparison of connected components found in a binary image using either N_4 or N_8 as structuring element. Every pixel corresponding to a certain set of connected components is labeled with the same number.

3.2.6 Pyramid reduction

Images generally contain much redundant information caused by correlation between neighbouring pixels. Pyramid reduction is a method used to simultaneously compress and low-pass filter an image, reducing noise and correlation between neighbouring pixels (Burt and Adelson, 1983). When properly used this will yield faster and more robust algorithms.

Pyramid reduction is an iterative process which, at every step, generates a filtered output image I_{n+1} half the size of the input image I_n in either direction. This process gives a pyramid structure of images where every new level is half the size of the previous level in either direction, often called an image pyramid, see figure 3.8. The initial input image is placed on pyramid level zero, $I_{in} = I_0$. After N iterations of this method the size of the final output image I_N will be $\frac{1}{2^N}$ of the original input image in either direction.

During every iteration a weighted sum of pixels from I_{n-1} is assigned to every pixel in I_n. The weighing function applied is similar to a Gaussian function. Thus this operation is similar to low-pass filtering and reduces noise in the image (Burt and Adelson, 1983).

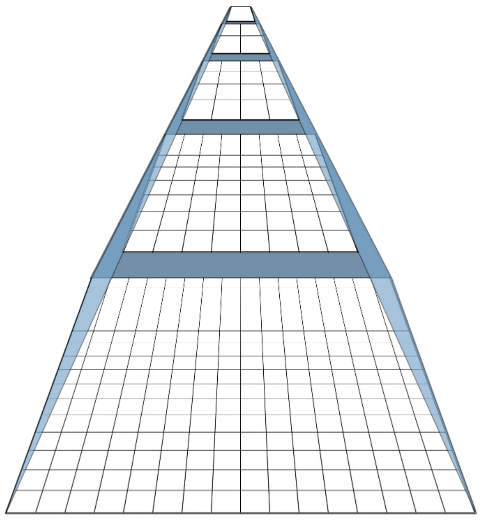


FIGURE 3.8: An example image of pyramid reduction. The bottom layer is the first input image, corresponding to pyramid level zero, which then iteratively has the length of its sides reduced to $\frac{16}{2^N}$ pixels at the higher pyramid levels.

3.3 Principal component analysis (PCA)

Principal component analysis (PCA) can be used to reduce the dimensionality of a data set with minimal loss of information. This is achieved by transforming the data set to a new set called the principal components, for which the first dimensions contain as much of the variation as possible. One can then consider a sufficiently large number of principal components and discard the rest, thus reducing dimensionality (Jolliffe, 2002).

The first principal component, PC_1, is chosen such that it can be used to explain as much of the variance in the data as possible. The second, PC_2, is then chosen as the direction orthogonal to PC_1 which explains as much of the remaining variance as possible. A general principal component PC_k where $k > 2$ is chosen such that it explains as much of the remaining variance in the data as possible while being orthogonal to $PC_1, ..., PC_{k-1}$ (Jolliffe, 2002).

Assume we have an observation matrix X with dimensions $n \times p$ and the mean of every column removed. Here n denotes the number of observations and p the number of dimensions. The sample covariance matrix S can then be calculated through (Jolliffe, 2002)

$$S = \frac{1}{(n-1)} X^T X \iff (n-1)S = X^T X. \tag{3.11}$$

The principal component scores can be found through the equation

$$Z = XV, \tag{3.12}$$

where Z is a matrix whose entry z_{ik} is the weight put on PC_k by observation i, and V is an orthogonal matrix whose columns are eigenvectors of S (Jolliffe, 2002). We now consider the singular value decomposition

$$X = U\Sigma V^T, \tag{3.13}$$

where X has rank r, U and V have orthonormal columns, U has dimensions $(n \times r)$ and V has dimensions $(p \times r)$. The matrix Σ is diagonal and $(r \times r)$ (Jolliffe, 2002).

Multiplying equation 3.13 on the right by V we get the equation

$$XV = U\Sigma \iff Z = U\Sigma, \tag{3.14}$$

and the principal component scores are thus given by

$$z_{ik} = u_{ik}\sigma_k^{1/2}, \tag{3.15}$$

where σ_k denotes the kth eigenvalue of $X^T X$. Letting γ_k denote the fraction of total variance explained by PC_k it is calculated as (Jolliffe, 2002; Navarra and Simoncini, 2010),

$$\gamma_k = \frac{\sigma_k^2}{\sum_{k=1}^{p} \sigma_k^2}, \tag{3.16}$$

The dimensionality reduction is finally performed by choosing m principal components to keep and discarding the rest. The data can be reconstructed using only these principal components as (Jolliffe, 2002)

$$\tilde{x}_{ij} = \sum_{k=1}^{m} u_{ik}\sigma_k^{1/2} v_{jk}, \tag{3.17}$$

where $\tilde{x}_{ij}$ denotes an element in the reconstructed data matrix $\tilde{X}$.

4 Methods

4.1 Data reduction

4.1.1 Pyramid reduction

In general the input images are large and contain much redundant information. To reduce the impact of noise and lower the computational cost of the algorithms the images are first compressed using pyramid reduction. Due to the standardized image acquisition process the features of interest in the input images are expected to be larger than 16×16 pixels. This allows for four steps of pyramid reduction. Using the top level of the image pyramid, level four, thus reduces the number of pixels handled by further algorithms to $\frac{1}{2^4} \cdot \frac{1}{2^4} = \frac{1}{256}$ of the original number.

To ensure that the pyramid reduction does not cause any issues or ambiguities the input images are cropped to have their height and width be multiples of $2^4 = 16$. Since the polymer samples never cover the entire width or height of the input images this can be done without loss of information. The pyramid reduction is then performed by the `impyramid` function in `MATLAB` (Burt and Adelson, 1983).

4.1.2 Pyramid expansion

It is preferable that the majority of the computations performed by further algorithms are performed on the top most level of the image pyramid to reduce computational effort and sensitivity to noise. To avoid losing information at higher resolutions every level of the image pyramid is saved. Once the algorithms operating at the top most pyramid level have finished corresponding results at lower pyramid levels are also computed.

Assume that a subset of pixels in the top level of the image pyramid, I_n, have been identified as an object, for example the colour line. For every pixel in that subset the four corresponding pixels on the lower pyramid level I_{n-1} are analyzed to conclude which of these are part of the object. The pixels in I_{n-1} corresponding to an interesting pixel in I_n are given by

$$\left\{ I_{n-1}(2x-1, 2y-1), I_{n-1}(2x-1, 2y), I_{n-1}(2x, 2y-1), I_{n-1}(2x, 2y) \right\} \sim I_n(x, y). \tag{4.1}$$

Of these four pixels those identified as part of the object are saved in a new subset at pyramid level $n-1$. This process is repeated at every pyramid level, ending with an object identification in the original image at level I_0.

Apart from reducing sensitivity to noise this process saves computational effort by drastically reducing the subset of interesting pixels to be analyzed at the lower levels. Note that it is only on the top pyramid level that all pixels of the image need to be analyzed. For the developed algorithms the four steps of pyramid reduction mean that I_4 has $\frac{1}{256}$ the number of pixels I_0 has. Thus every algorithm working on the top level of the image pyramid will be substantially faster.

4.2 Colour segmentation methods

The red ink used to colour regions of the samples can be seen at the top of figure 4.1a. Splitting this image into its three colour channels and looking at their intensities separately allows for an interesting observation. Despite the ink being red it creates the biggest intensity difference in the blue and green colour channels. This can be seen comparing figures 4.1b - 4.1d.

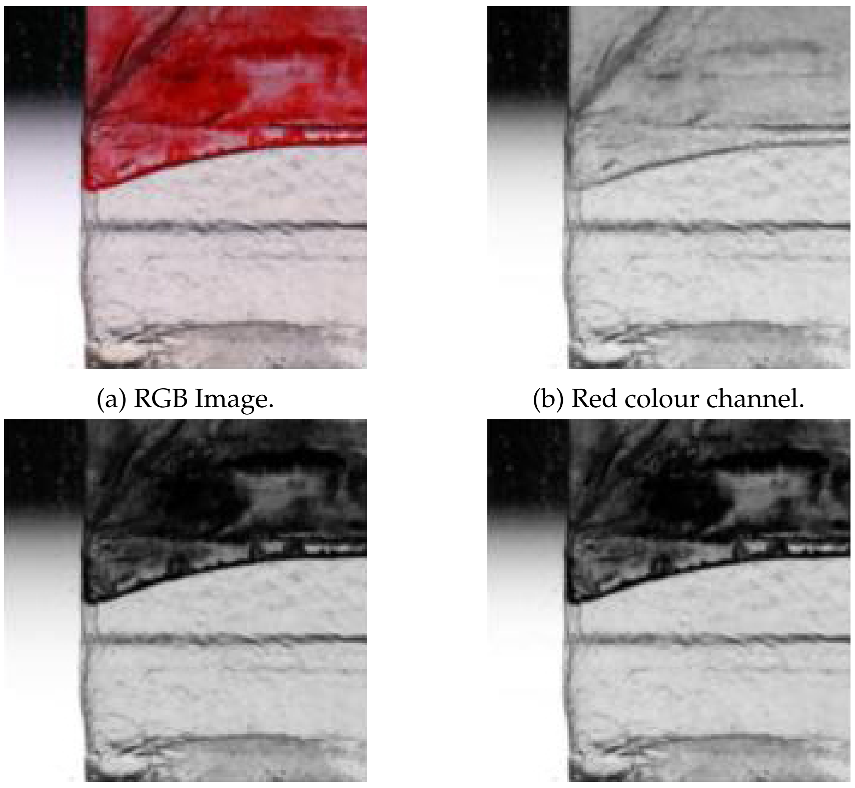

(a) RGB Image.
(b) Red colour channel.
(c) Green colour channel.
(d) Blue colour channel.

FIGURE 4.1: The RGB representation of a sample and the intensities in its three colour channels separately. In the colour channel images brighter pixels indicate higher intensities and darker pixels indicate lower intensities.

4.2.1 RGB and PCA

Knowing the red ink is most apparent in the blue and green colour channels one could choose to work with these to find the red ink in the image. Another option is to use PCA in an effort to single out the ink in the image. Since the input images are composed mainly of polymer, a light source in the background and the ink it turns out that the majority of the image will have relatively equal intensities in all three colour channels. The pixels where there is red ink will however greatly deviate from this pattern. This can be seen looking at figures 4.1b-4.1d, where the pixels in the three colour channels corresponding to the red ink have varying intensity and the pixels not corresponding to the red ink have relatively similar intensities.

Performing a PCA yields a first principal component (PC_1) with strong signal where there is no red ink in the image. The second principal component (PC_2) explains the variation caused by the ink. The third principal component (PC_3) is discarded since it has very low impact and can be considered as noise. Principal components PC_1 and PC_2 can be seen in figures 4.2b and 4.2c respectively. The corresponding RGB representation can be seen in figure 4.2a.

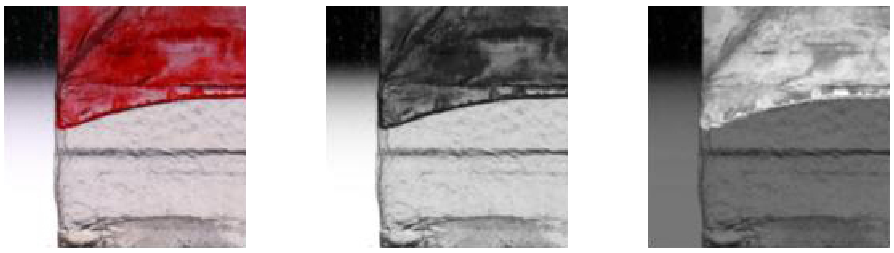

(a) RGB representation. (b) PC_1. (c) PC_2.

FIGURE 4.2: The two relevant principal components resulting from PCA of one of the samples. Brighter pixels indicate higher intensities and darker pixels indicate lower intensities.

Further algorithms developed in this Thesis will generally use the first two principal components instead of some combination of the red, green and blue colour channels. This is done since the composition of the input images allows PCA to describe the red ink with its second principal component. An additional benefit of using the principal components is that future variations of ink colour should not matter for the algorithms assuming the new colour does not resemble the colour of the polymer.

4.3 Sample border algorithms

The samples do no cover the entire input images, and therefore it is useful to limit the image analysis to the pixels which constitute the sample. Doing this ensures that no computational effort is wasted trying to find features where there is no sample. To avoid analyzing unnecessary pixels we are interested in finding the sample edges in the image and disregard pixels outside the sample in subsequent algorithms.

The majority of every sample is illuminated by a background light source, and therefore the most prevalent sample border will be that between polymer and light source. There is, however, no light source in the top part of the images and therefore a different method is needed there. Finally part of the sample is coloured by red ink, and there we use colour segmentation to find the sample border.

The area of the image containing the sample then lies between the leftmost and rightmost columns containing the sample for any of the methods. The sample begins at the topmost row of the image and ends at the border between polymer and light source. Pixels not part of this area are disregarded before further processing.

4.3.1 Border between polymer and light source

The edge between the polymer and the light source is easily detected using either PC_1 or a grayscale conversion of the RGB image. The grayscale conversion is done using the `MATLAB` function `rgb2gray` assigning to every pixel its luminance see, (International Telecommunication Union, 2011), for details. The principal component analysis is performed as previously described. As you can see comparing figures

4.3b and 4.3a, PC_1 and the grayscale representation differ very little. This is contrasted by PC_2 in figure 4.3c. The lack of red ink causes pixels in PC_2 to have low intensities.

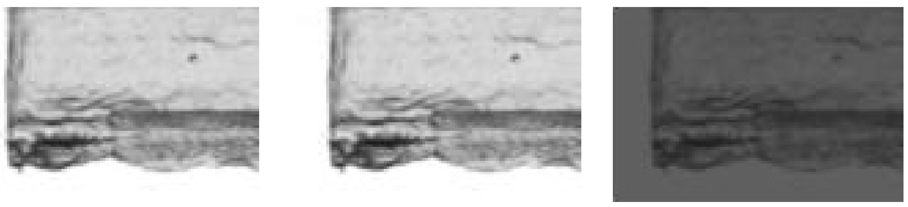

(a) Grayscale representation. (b) PC_1. (c) PC_2.

FIGURE 4.3: Comparison between the grayscale representation, PC_1 and PC_2 for border detection between polymer and light-source.

Simply thresholding either the grayscale image or PC_1 using a properly chosen threshold will yield good results of which pixels correspond to the light source and which correspond to the polymer.

To calculate the threshold we first consider a portion of the image with pixels known to contain only light source. This can be done because of the consistent composition of all input images. Denoting this portion of the image as I_{ls} and assuming a normal distribution for the pixel intensities the threshold is calculated as

$$\mu_{ls} = \frac{1}{n}\sum_{i=1}^{n} p_i, \tag{4.2}$$

$$\sigma_{ls} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - \mu_{ls})^2}, \tag{4.3}$$

$$T_{ls} = \mu_{ls} - 3 \cdot \sigma_{ls}, \tag{4.4}$$

where n denotes the number of pixels in I_{ls} and p_i is a pixel in that part of the image. The polymer is defined to be the pixels lower than the threshold T_{ls}, ensuring they are not part of the light source with approximately $99,7\%$ confidence. Naturally other confidence levels could be used if they are more suitable, for example 99% confidence achieved by replacing 3 with 2.576 in equation 4.4.

The binary image resulting from thresholding PC_1 can be seen in figure 4.4a. Defining the white pixels with intensity one of the thresholded image as polymer we end up with the border shown as a red line in figure 4.4b.

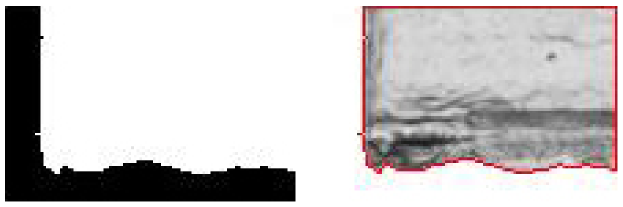

(a) Binary image. (b) Polymer border.

FIGURE 4.4: The left subfigure shows the binary image resulting from thresholding figure 4.3b and the right image shows the border consisting of the outermost pixels of the connected component corresponding to the polymer.

4.3.2 Border between polymer and unilluminated background

The case where there is no light source in the background is solved in a similar manner as the case with a light source in the background. Since there is no red ink at the border we note that the difference between PC_1 and the grayscale representation is once again small, and that PC_2 takes very different values. The same thresholding method should therefore work well if the threshold is calculated using a part of the image known to contain unilluminated background. The difference between the grayscale representation, PC_1 and PC_2 can be seen in figure 4.5.

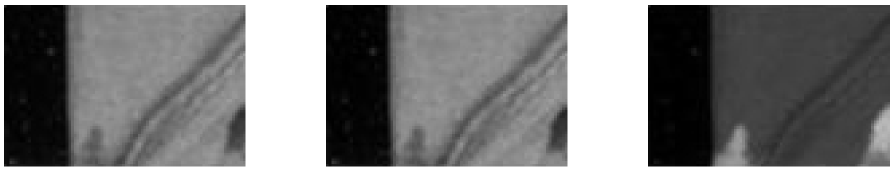

(a) Grayscale representation. (b) PC_1. (c) PC_2.

FIGURE 4.5: Comparison between the grayscale representation, PC_1 and PC_2 for border detection between polymer and unilluminated background.

We use PC_1 and consider a portion of the image known to contain pixels belonging to the unilluminated background as we calculate the threshold. Denoting this portion of the image I_{ub} and assuming the pixel intensities are normally distributed lets us calculate the threshold as

$$T_{ub} = \mu_{ub} + 3 \cdot \sigma_{ub}, \tag{4.5}$$

where μ_{ub} and σ_{ub} are calculated like their corresponding parameters in equations 4.2 and 4.3, except with pixels from I_{ub}. The resulting border when thresholding with T_{ub} and choosing the pixels larger than the threshold can be seen as the red line in figure 4.6b. The binary thresholded image can be seen in figure 4.6a.

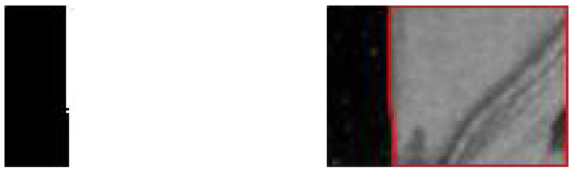

(a) Binary image. (b) Polymer border.

FIGURE 4.6: The border of the polymer against an unilluminated background. The left subfigure shows the binary image resulting from the thresholding operation and the right subfigure shows the border as a red line consisting of the outermost pixels of the connected component corresponding to the polymer.

4.3.3 Border between red ink and mixed lighting background

At certain locations in the image the sample meets the transition between light source and unilluminated background. We call this mixed lighting and to handle this case a composite image is created. It is done by first normalizing the values of PC_1 and PC_2 to lie between zero and one, and then creating a new image by assigning a new value to every pixel according to

$$I_{composite} = I_{PC2} \cdot (1 - I_{PC1}). \tag{4.6}$$

Here $\cdot$ denotes an element wise multiplication and $I_{composite}$ is the new image. The composite image is created to suppress the intensity of both the light source and the unilluminated background. The second principal component PC_2 will have intensities close to zero where there is unilluminated background and the inverse of the first principal component $(1 - PC_1)$ will have intensities close to zero where there is light source. The red ink will have relatively high intensities in both images, and hence the composite image enhances the red ink compared to the background, see figure 4.7d.

To choose a threshold for the composite image we consider the area of transition between unilluminated background and light source. Choosing a portion of $I_{composite}$ containing those pixels and denoting it I_{mix} we calculate the threshold in a similar manner to T_{ls} and T_{ub}.

$$\mu_{mix} = \frac{1}{n} \sum_{i=1}^{n} p_i, \tag{4.7}$$

$$\sigma_{mix} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (p_i - \mu_{mix})^2}, \tag{4.8}$$

$$T_{mix} = \mu_{mix} + 3 \cdot \sigma_{mix}, \tag{4.9}$$

where n denotes the number of pixels in I_{mix} and p_i are pixels in that portion of the composite image. A binary image is created by thresholding and defining the pixels with intensities larger than T_{mix} to be red ink.

A sample where there is red ink bordering both light source and unilluminated background can be seen in figure 4.7a. Its first and second principal components can be seen in figures 4.7b and 4.7c. The described composite image can be seen in figure 4.7d and the thresholded binary image is seen in figure 4.7e. Finally the resulting red ink border can be seen as the red line in figure 4.7f.

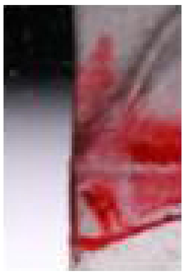
4.4 Colour line algorithm

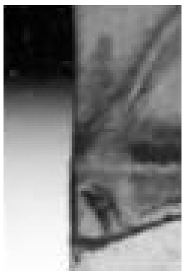
4.4.1 Thresholding and morphology

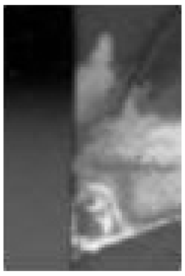
To find the pixels coloured by red ink we use the threshold T_{mix} on the composite image $I_{composite}$. While this operation does find the pixels containing red ink it turns out this is not enough to locate the colour line. The process in which the colour is applied is not very precise and the colour is unevenly distributed, which the algorithm has to account for it. The sought edge is a slightly darker border found between the red ink region and the polymer.

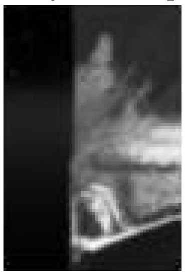
An example of the uneven distribution of red ink as well as the slightly darker edge between red ink and polymer can be seen in figure 4.8a. A manually drawn approximate colour line can be seen in figure 4.8b.

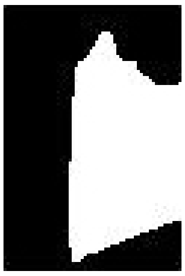
Applying the threshold T_{mix} calculated as in equation 4.9 to the composite image $I_{composite}$ defined as in equation 4.6 yields a binary image. Because of the uneven distribution of red ink further steps are needed to find the colour line. To account for small patches of missing red ink we apply morphological closing. In preparation for closing very small connected components, most likely either speckles of red ink or pixels mistaken for red ink, are removed. This is done because those might otherwise


(a) Polymer sample.


(b) PC_1.


(c) PC_2.


(d) Composite image of PC_1 and PC_2.


(e) Binary image.

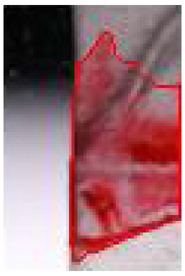

(f) Sample with detected border.

FIGURE 4.7: The subfigures show the procedure of detecting the border of red ink against both light source and unilluminated background. The first subfigure shows the sample, the second and third show PC_1 and PC_2, the fourth a composite image using PC_1 and PC_2, the fifth shows the binary image resulting from thresholding the composite image and the sixth subfigure shows the detected border as a red line.

be included in the connected component describing the red ink region after closing, causing unwanted errors. Connected components smaller than a certain size are removed. The size has been manually chosen after looking at the input images in the modelling data set.

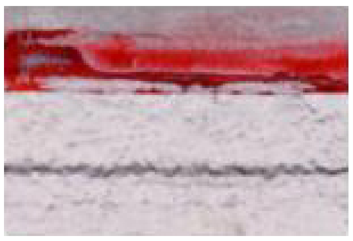
The first principal component corresponding to figure 4.8a can be seen in figure 4.9a, and the second in figure 4.9b. The composite image can be seen in figure 4.9c. The binary image resulting from thresholding with T_{mix} can be seen in figure 4.9d.

4.4.2 Error correction

Since the bottom of the red ink region in figure 4.9d does not match the expected colour line in figure 4.8b further steps are needed. The assumption that the colour line is continuous is used to create an error correction algorithm. Looking at the bottom of the connected components corresponding to the red ink in figure 4.9d we seek discontinuities in the y-coordinate.

The sought colour line should take exactly one y-value for every x-value and the continuity ensures that for any pixel $p_i(x_i, y_i)$ which is not the first nor the last pixel in the line, its neighbours in columns x_{i-1} and x_{i+1} can be found in $N_8(p_i(x_i, y_i))$, see figure 4.10.

To find discontinuities we look for larger positive or negative gradients than is expected of a continuous line. Locations with negative gradient discontinuities as well as positive gradient discontinuities are saved separately. We expect every error to be represented by a pair of one negative and one positive discontinuity.


(a) Sample with missing red ink.

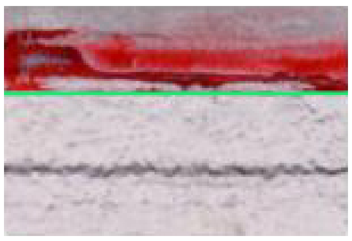
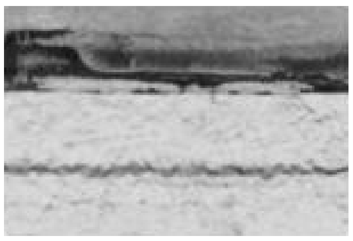
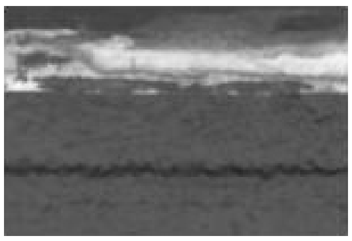
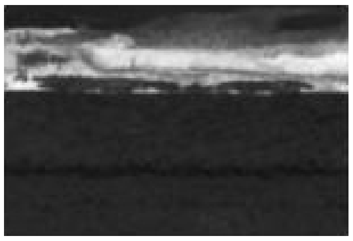

(b) Approximate sought colour line

FIGURE 4.8: The first subfigure shows a polymer samples with missing red ink. The second subfigure shows the approximate sought colour line as a green line.


(a) PC_1.


(b) PC_2.


(c) $I_{composite}$.

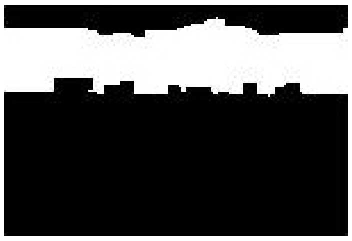

(d) Binary image.

FIGURE 4.9: The subfigures depict the sample in figure 4.8a. The first subfigure shows PC_1, the second subfigure shows PC_2 and the third subfigure shows $I_{composite}$. Finally the fourth subfigure shows the binary image resulting from thresholding.

Once the discontinuities have been identified the errors are dealt with one by one in an iterative manner. During an iteration an error is chosen as the closest occurrence of a discontinuity pair in the image. The area between those discontinuities is considered faulty. The area is then extended at either side until the furthest discontinuity of the same type (negative/positive) without encountering a discontinuity of the other type is found. Finally the line in the error area is updated while remaining continuous with the rest of the line. This process is repeated until no error pair within a manually chosen distance is found.

The maximum distance between an error pair has been manually chosen by looking at the results for various settings using the modelling data set.

If we consider the binary image in figure 4.9d and locate the discontinuities we end up with figure 4.11a where the negative gradient discontinuities are marked as red dots and the positive as green dots. One of the error correction iterations would start by choosing the discontinuity pair labeled 1, after which the area would be extended to the right until the positive discontinuity labeled 2 was reached. The iteration would then update the line between the negative discontinuity labeled 1 and

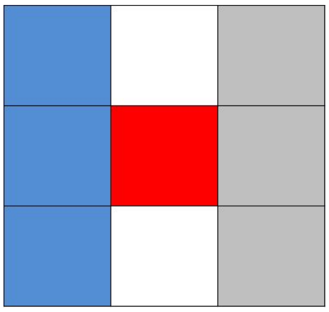
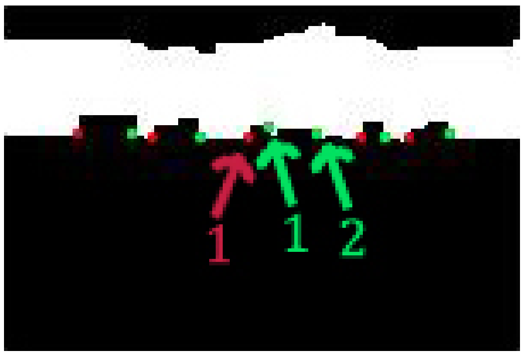
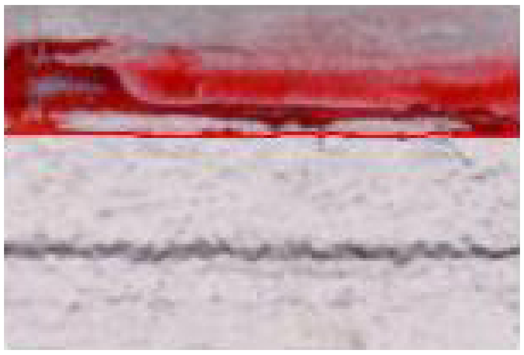


FIGURE 4.10: The eight neighbourhood of a pixel p_i which is marked as red. If p_i is part of a continuous line the previous pixel p_{i-1} is expected to be found as one of the blue pixels and the next pixel p_{i+1} is expected to be found as one of the gray pixels.

the positive discontinuity labeled 2. The results of applying iterative error correction with the algorithm described above can be seen in figure 4.11b, where the resulting colour line is marked as a red line.



(a) Identified discontinuities.

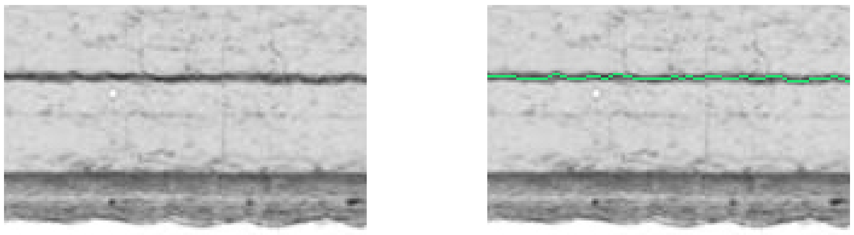


(b) Corrected colour line.

FIGURE 4.11: The first subfigure shows the identified gradient discontinuities. The negative discontinuities are marked as red dots and the positive discontinuities are marked as green dots. In the second subfigure the corrected colour line is marked as a red line.

4.5 Shadow line algorithm

The second line of great interest in the polymer samples is the shadow line. It is a horizontally continuous line in the images which separates two regions with similar textures, namely plain polymer without any added ink. The algorithms developed use the first principal component, although the grayscale representation would also work. An example of what the shadow line looks like can be seen in figure 4.12a. Manually painting one pixel in every column green at the approximate location of the shadow line yields the green line in figure 4.12b.



(a) PC_1. (b) Sought shadow line in green.

FIGURE 4.12: Both subfigures show PC_1 of a sample. The second subfigure has the sought shadow line manually marked with green pixels.

We want to automatically detect the shadow line in figure 4.12b and note that it is darker than the pixels in a small neighbourhood around it. Therefore the intensity of a pixel in PC_1 can be used to determine whether it is likely or not it is part of the shadow line. We also note that the intensity difference in PC_1 is rather abrupt and hence the absolute value of the gradient should be large at the top and the bottom of the line. We now want to use this knowledge to locate the top of the shadow line as well as the bottom of the shadow line and then define the shadow line as the line centered between those.

4.5.1 Location likelihood

The polymer samples are manufactured in such a way that great similarities can be found between every sample. Therefore it is useful to apply a filter narrowing down the number of pixels where we search for the shadow line. Accounting for some variance this essentially boils down to assigning a likelihood to the location of every pixel in the image. It is sufficient to assign a likelihood to the location in every column containing the shadow line.

To do this we first calculate an approximate location of the shadow line in the image and then assume the distance to the true shadow line is well described by a normal distribution. The approximate shadow line, L_{approx}, is calculated by fitting a second order polynomial to a line created by simply choosing pixels a set number of pixels above the bottom of the polymer sample. The number of pixels one should take depends on the machine settings used as the sample was manufactured.

The likelihood that a pixel belongs to the sought line is then given by a normal distribution and its distance from the approximate shadow line. The expected distance is naturally chosen to be $\mu = 0$ and the variance σ^2 has been chosen manually by considering the modelling data set.

The location likelihood image is calculated by assigning to every pixel the likelihood of being located at its distance from L_{approx} and then normalizing the image to take values between 0 and 1. The image is saved as I_{Loc}.

The approximate shadow line a set number of pixels above the bottom of the polymer sample can be seen as the blue line in figure 4.13b. The corresponding second order polynomial can be seen as the red line in the same figure. The location likelihood image I_{Loc} can be seen in figure 4.13c. These figures correspond to the sample in figure 4.13a.

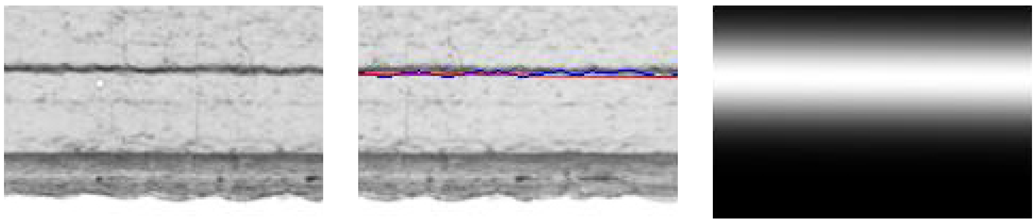

(a) PC_1. (b) Approximate shadow line. (c) I_{Loc} likelihood image.

FIGURE 4.13: The first subfigure shows PC_1 of a sample. The second subfigure shows the approximate shadow line as a blue line. Additionally it shows the second order polynomial approximating the shadow line location in red. Finally the third subfigure shows the location likelihood image I_{Loc}.

4.5.2 Intensity likelihood

In figure 4.12 we see that the sought line is darker than the surrounding pixels and thus has lower intensity. The intensity of the line varies both within a single sample as well as between samples. It will, however, always have lower intensity than its surrounding pixels. If we describe the likelihood of a certain intensity belonging to the shadow line as a normal distribution it is possible to create a likelihood image where every pixel is assigned a new value based on its intensity is PC_1.

To calculate the expected value μ_{Int} and the standard deviation σ_{Int} an approximate set of pixels belonging to the shadow line is chosen as the darkest pixels within a manually entered maximum distance from L_{approx}. This yields a set of pixels where most but not all belong to the shadow line. The sought parameters are then calculated using the standard equations for expected value and standard deviation, see equations 4.2 and 4.3.

Once the likelihood image, I_{int}, has been created it is normalized to take values between zero and one. The likelihood image corresponding to figure 4.14a can be seen in figure 4.14b.

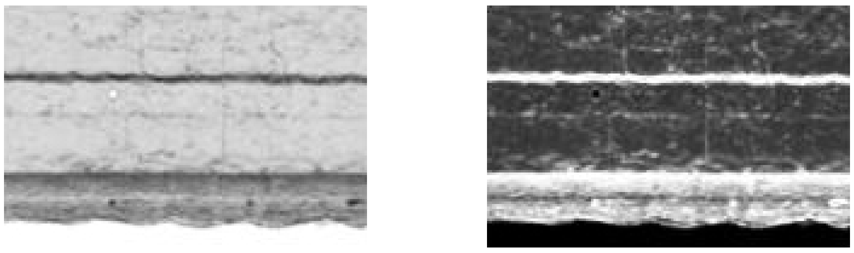

(a) PC_1. (b) I_{Int} likelihood image.

FIGURE 4.14: The first subfigure shows the intensities in the first principal component of a sample. The second shows the corresponding likelihood of those pixels belonging to the shadow line, where brighter pixels are more likely.

4.5.3 Gradient likelihood

To find the top and bottom of the shadow line we consider the gradient of PC_1 calculated using a Prewitt gradient operator mask. This yields the magnitude and direction of the gradient at every pixel considering its eight neighbourhood N_8. Assuming the gradient magnitude along the shadow line is normally distributed we want to find the expected gradient magnitude μ_{Mag} and corresponding standard deviation σ_{Mag}.

The gradient direction for pixels belonging to the shadow line is assumed to be normally distributed. The parameters differ between the top and the bottom of the shadow line and have been manually chosen. The chosen values have been tested on the modelling data set and are

$$\begin{aligned} \mu_{TopDir} &= 90 & \sigma_{TopDir} &= 75 \\ \mu_{BtmDir} &= -90 & \sigma_{BtmDir} &= 75 \end{aligned}$$

where $TopDir$ denotes the gradient direction parameters belonging to the top of the shadow line and $BtmDir$ the bottom of the shadow line.

For the parameters corresponding to the gradient magnitude a data set of pixels in the gradient magnitude image are chosen as the pixels in an area around L_{approx} with highest gradient magnitude in their column. The sought parameters μ_{Mag} and σ_{Mag} are then calculated using the standard equations for expected value and standard deviation, see equations 4.2 and 4.3.

$$\mu_{Mag} = \frac{1}{n}\sum_{i=1}^{n} p_i, \tag{4.10}$$

$$\sigma_{Mag} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - \mu_{mix})^2}, \tag{4.11}$$

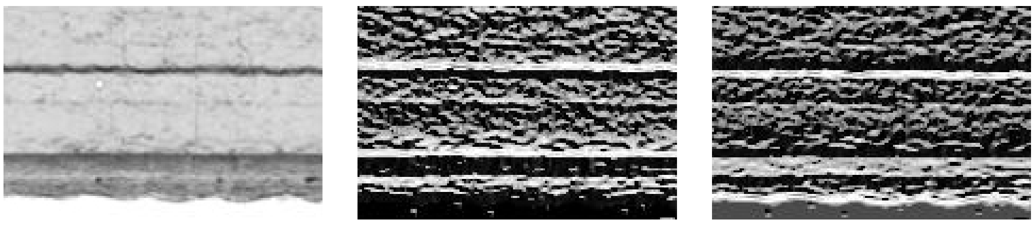
where n denotes the number of pixels p_i in the set of gradient magnitude pixels.

Creating a new likelihood image for each normal distribution and normalizing those to have values between zero and one gives the likelihood images I_{Mag} describing the likelihood of the gradient magnitude as well as I_{TopDir} and I_{BtmDir} describing the likelihood of the gradient direction. The likelihood that a single pixel belongs to either the top or the bottom of the shadow line is then calculated by multiplying the two relevant images. This gives the new likelihood images

$$I_{Top} = I_{Mag} \cdot I_{TopDir}, \tag{4.12}$$

$$I_{Btm} = I_{Mag} \cdot I_{BtmDir}, \tag{4.13}$$

where $\cdot$ denotes element wise multiplication. This means that the distributions corresponding to the direction and magnitude of the gradient are treated as independent distributions. Looking at figure 4.15b we see the likelihood that the pixels in figure 4.15a belong to the top of the shadow line. The corresponding image for the bottom of the shadow line is seen in figure 4.15c.



(a) PC_1. (b) I_{Top} likelihood image. (c) I_{Btm} likelihood image.

FIGURE 4.15: The first subfigure shows the first principal component for a sample. The second and third subfigures show the likelihood that a given pixel belongs to the top or the bottom of the shadow line, I_{Top} and I_{Btm}.

4.5.4 Combined likelihood

To find the top and the bottom of the shadow line we seek to combine the likelihood images given by the intensity, gradient and location, I_{Int}, I_{Top}, I_{Btm} and I_{Loc}. These normal distributions are treated as independent and are combined by element wise multiplication. Calling I_{Loc} a location filter we consider the likelihood of a pixel belonging to the top of the shadow line with or without location filter and do the same for the bottom of the shadow line. This gives the following likelihood images

$$I_{LHTopLoc} = I_{Int} \cdot I_{Top} \cdot I_{Loc}, \tag{4.14}$$

$$I_{LHTop} = I_{Int} \cdot I_{Top}, \tag{4.15}$$

$$I_{LHBtmLoc} = I_{Int} \cdot I_{Btm} \cdot I_{Loc}, \tag{4.16}$$

$$I_{LHBtm} = I_{Int} \cdot I_{Btm}. \tag{4.17}$$

The likelihood images corresponding to the top of the shadow line with and without location filter are denoted $I_{LHTopLoc}$ and I_{LHTop}. The corresponding likelihood images for the bottom of the shadow line are $I_{LHBtmLoc}$ and I_{LHBtm}.

The four likelihood images can be seen in figure 4.16. Subfigure 4.16a corresponds to $I_{LHTopLoc}$, 4.16b to $I_{LHBtmLoc}$, 4.16c to I_{LHTop} and finally 4.16d to I_{LHBtm}.

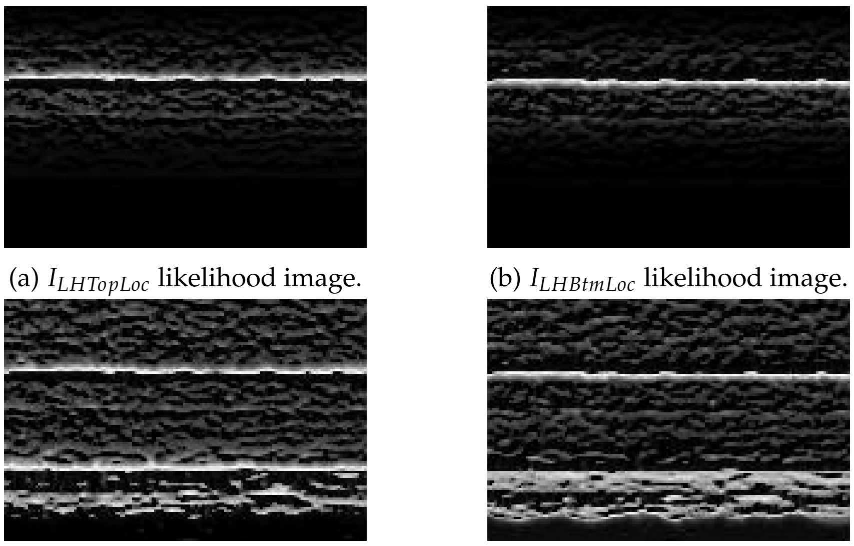
4.5.5 Most likely continuous shadow line

Given a likelihood image the next step is to extract the continuous shadow line. The continuity ensures that every pixel in the shadow line will have exactly one other pixel belonging to the line in the left column of its eight neighbourhood, N_8, and one in the right column of its N_8, see figure 4.10. This is true assuming the pixel does not lie at the start or end of the line in which case there will only be a pixel in either the right or the left column of its N_8. Defining the shadow line as the set of pixels $L = \{p_1, p_2, ..., p_l\}$, where l denotes the number of pixels in the line we get the following relations

$$p_i \in N_{8(:,1)}(p_{i+1}), \; \forall p_i, i < l, \tag{4.18}$$

$$p_i \in N_{8(:,3)}(p_{i-1}), \; \forall p_i, i > 1. \tag{4.19}$$

Here $N_{8(:,1)}$ denotes the first column of the eight neighbourhood and $N_{8(:,3)}$ denotes the third column of the eight neighbourhood.



(a) $I_{LHTopLoc}$ likelihood image. (b) $I_{LHBtmLoc}$ likelihood image.

(c) I_{LHTop} likelihood image. (d) I_{LHBtm} likelihood image.

FIGURE 4.16: The combined likelihood images showing which pixels in the image are likely to belong to the shadow line. Brighter pixels mean higher likelihood. The four variations show the likelihood of a pixel belonging to the top or the bottom of the shadow line, either with or without a location filter applied.

4.5.5.1 Likelihood summed column-wise

Given a likelihood image a first approach is to extract the shadow line as the continuous line with the highest summed likelihood. This approach allows us to extract the shadow line by following some fast and easy steps.

First we define two matrices with sizes identical to the image, M_{SL} to hold the summed likelihood at every location and M_{PPL} to hold the previous pixel locations. Choosing the left most column of the sample in the input image as a starting point we then iteratively treat one column at a time until the right most column of the sample is reached.

Denoting the starting column x_1 and the final column x_n, we first enter the likelihood values from column x_1 of the used likelihood image, I_{LH}, into column x_1 of M_{SL}. Then for $i > 1$

$$M_{SL}(x_i, y) = max(M_{SL}(x_i - 1, y - 1), M_{SL}(x_i - 1, y), M_{SL}(x_i - 1, y + 1)) + I_{LH}(x, y), \tag{4.20}$$

and $M_{PPL}(x_i, y)$ is assigned the location of the pixel chosen in the max expression. This is calculated for $y = 1, 2, ..., M$ where M is the number of rows in the image. The process is repeated until x_n has been reached.

The resulting summed likelihood matrix M_{SL} will naturally have higher entries in columns to the right. The final path is calculated by locating the pixel with the highest summed likelihood in column x_n and retracing its path by iteratively looking at which pixel preceded it in M_{PPL} until the first pixel is reached. Denoting the found line L we get the final pixel of it as

$$p_n = max(p(x_n, y), y = 1, 2, ..., M) \in L, \tag{4.21}$$

and the other pixels by iteratively retracing the path through

$$p_{n-i} = M_{PPL}(p_n) \in L. \tag{4.22}$$

This process is repeated for $i = 1, 2, ..., n-1$, which is when the first column of the sample is reached. The resulting line when applying this algorithm to $I_{LHTopLoc}$ can be seen in figure 4.17b and the corresponding results for I_{LHTop} can be seen in figure 4.17c. The corresponding image of PC_1 can be seen in figure 4.17a.

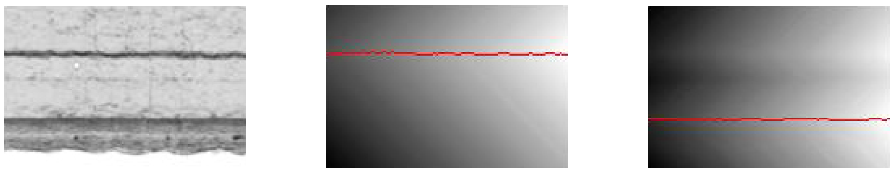

(a) PC_1. (b) M_{SL} for $I_{LHTopLoc}$. (c) M_{SL} for I_{LHTop}.

FIGURE 4.17: The normalized summed likelihood matrices where brighter pixels have higher likelihood. We see and compare with as well as without location filter and note that removing the location filter with this approach results in an error. The red lines indicate L, the shadow line found by the algorithm.

4.5.5.2 Likelihood summed over segments

Given the results in figure 4.17c we reconsider our assumptions regarding the shadow line. It is not necessarily the most likely continuous line over the entire image, instead it is considered to be the topmost continuous line in the image with summed likelihood larger than a threshold.

Noise in the image can be assigned high likelihood by the algorithm. To avoid errors the shadow line is considered to be the most likely continuous line from a pixel of the line to the last column of a short segment of the image. This reduces the impact of noise since it ensures that the shadow line does not take detours across less likely pixels to objects farther away than the width of the segments.

We now split the likelihood image I_{LH} into segments with 50% overlap. For every segment the most likely paths from the first column to the last column are saved as ones in the binary path matrix M_P. Those paths are given by the local maximums in the final column. The overlap ensures that there are no discontinuities between the paths. Given the binary path matrix we define a new likelihood image through

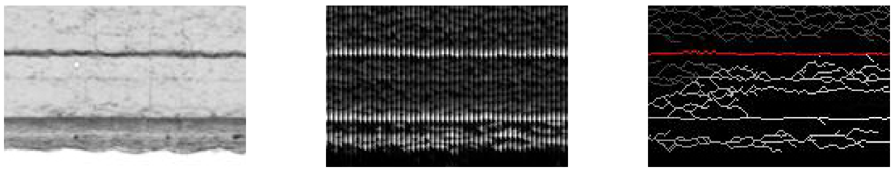
$$I'_{LH} = I_{LH} \cdot M_P \tag{4.23}$$

We then define a summed likelihood matrix corresponding to the new likelihood image, M'_{SL}, as well as a new matrix where previous pixel locations are stored, M'_{PPL}. The process described in the previous section is then performed to calculate the most likely continuous lines in I'_{LH}.

We again denote the first column where the shadow line is found as x_1 and the last as x_n. The shadow line, L, is then defined to be the top most element in column x_n of M'_{SL} with summed likelihood larger than a manually chosen threshold. The threshold has been chosen considering the results from the modelling data set. The shadow line is then found as in equation 4.22 using M'_{PPL} instead of M_{PPL}.

The sample in figure 4.17c is the same sample as in figure 4.18a. We see that the algorithm summing likelihood over segments successfully isolates the two most likely paths and chooses the topmost path. The summed likelihood of the segments,

M_{SL}, can be seen in figure 4.18b. The new summed likelihood matrix M'_{SL} and the detected path in red can be seen in figure 4.18c.



(a) PC_1.

(b) M_{SL} for every segment.

(c) M'_{SL} with found shadow line in red.

FIGURE 4.18: The first subfigure shows PC_1 of a sample. The second subfigure shows the summed likelihood matrix M_{SL} when calculated for many segments of the image I_{LH}. The third subfigure shows the new summed likelihood matrix M'_{SL} as well as the detected shadow line in red.

4.5.6 Removal of irrelevant shadow line pixels

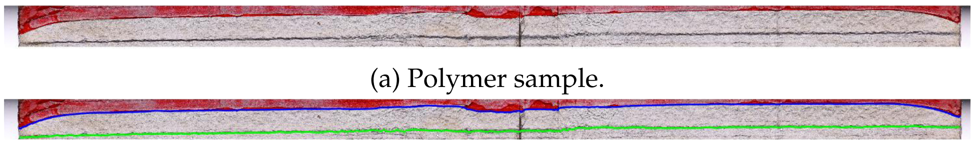
For some input images the shadow line is not expected to begin at the border of the sample but rather a few pixels in. The missing pixels depict polymer when this is the case, and therefore the likelihood in I_{LH} that they belong to the shadow line should be small.

To remove the pixels in L which should not be included in the shadow line set we consider the likelihood that the pixels at either end of the shadow line belongs to it. Pixels are removed from the start of the shadow line until the first pixel with likelihood larger than a threshold is found, $I_{LH}(p_i) > T_{Rem}$. This pixel is chosen as the new starting point of the shadow line. The corresponding process yielding a new end point of the shadow line is also performed.

The threshold T_{Rem} has been chosen manually by looking at shadow lines in the modelling data set images.

4.6 Sample differences by machine settings

It is interesting to evaluate whether the three different machine settings cause differences in the acquired samples. We are interested in the shape and size of the area between the two extracted lines, an example of which can be seen in figure 4.19. Subfigure 4.19a shows the examined sample. Subfigure 4.19b shows the colour line in blue and the shadow line in green.



(a) Polymer sample.

(b) Colour line marked in blue and shadow line marked in green.

FIGURE 4.19: Two subfigures depicting the same polymer sample. The second subfigure shows the colour line as a blue line and the shadow line as a green line.

In general the number of pixels in an input image varies, and so does the location of the two lines. Therefore we first compute the distance between the lines for each column in an image, resulting in a vector of length n. To compare data between images for which the length of the lines varies slightly, the distances are linearly interpolated to a common resolution of m columns per image.

The distance between the lines is saved as columns in an $m \times s$ matrix M_L, where s denotes the number of samples. To compare low, medium and high machine settings the columns corresponding to different machine settings are noted. The mean value of the the matrix is then subtracted and saved as a variable, after which principal component analysis is performed on the matrix to analyze the area between the two lines and its dependency on machine settings.

5 Results

In this chapter algorithm results from the validation data are presented. The first section presents thresholds automatically calculated to detect the sample borders. The second section describes the automatically calculated parameters with which the shadow line likelihood images are calculated. The third section presents a survey about the quality of the two extracted lines. The shadow line has been extracted using likelihood summed over segments. Finally the last section presents results of the principal component analysis performed on M_L, separated for the different machine settings when possible.

5.1 Sample border thresholds

To find the borders of the sample in an input image three thresholds are automatically calculated considering different parts of the image. The three thresholds are T_{ls}, above which pixels in PC_1 describe the light source, T_{ub}, below which pixels in PC_1 constitute the unilluminated background and finally T_{mix}, above which pixels in $I_{composite}$ depict red ink.

Boxplots of the three thresholds calculated for the 30 images in the validation data set can be seen in figure 5.1. The more interested reader can see the corresponding table in appendix A table A.1.

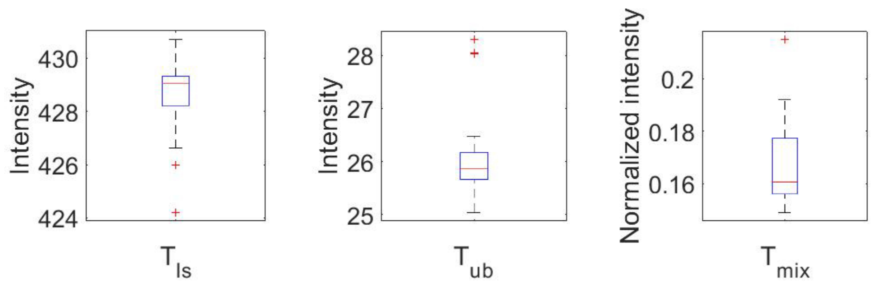


FIGURE 5.1: The three different thresholds automatically calculated to detect the sample. The two thresholds T_{ls} and T_{ub} describing the light source and unilluminated background in PC_1, and T_{mix} describing the red ink in $I_{composite}$.

5.2 Shadow line likelihoods

The algorithm extracting the shadow line calculates two likelihood images, I_{Int} and I_{Mag}. These likelihood images are created using normal distributions and the parameters are automatically evaluated by considering an approximation of the shadow line based on the bottom of the sample.

For every sample four parameters are calculated. Those are the expected values of the intensity and gradient magnitude, μ_{Int} and μ_{Mag}, as well as the corresponding standard deviations, σ_{Int} and σ_{Mag}. The extracted values from the validation data set can be seen in figure 5.2. The interested reader can get more detailed information in appendix A table A.2.

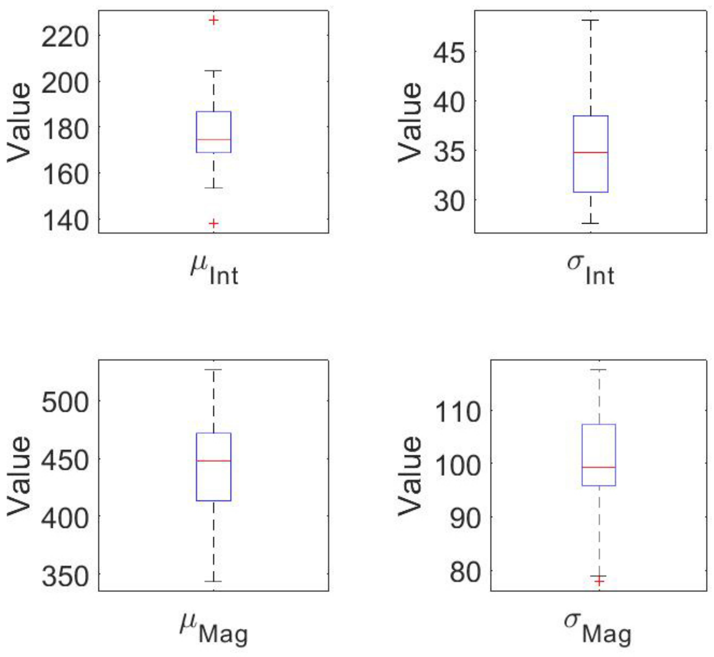


FIGURE 5.2: The four parameters with which the two likelihood images I_{Int} and I_{Mag} are created. The values are extracted from the validation data set. The $\mu-$parameters describe the expected values and the $\sigma-$parameters the standard deviations.

5.3 Survey: Quality of extracted lines

There is no precise knowledge about which pixels constitute the sought lines before the algorithms find them, and therefore it is difficult to evaluate the quality of the results. This subsection contains a small survey where four Tetra Pak® employees gave their opinion of the results according to a grading scale. In the following grading scale further evaluation refers to a quality rating of the sample,

5 The extracted lines perfectly or almost perfectly match the true lines.

4 There are minor errors which likely have no or very small impact on further evaluation.

3 There are errors which will likely moderately impact further evaluation.

2 Errors are likely to cause bad results for further evaluation.

1 The extracted lines are very bad and further evaluation is rendered useless.

The results of the survey can be sen in figure 5.3, where the answers from an employee are shown as filled dots of a certain colour. The mean grading scores are marked as black circles. We note that the results are generally good and that only four samples have a mean grading score of less than 4. A reader who prefers to see this as a table is referred to appendix A table A.3.

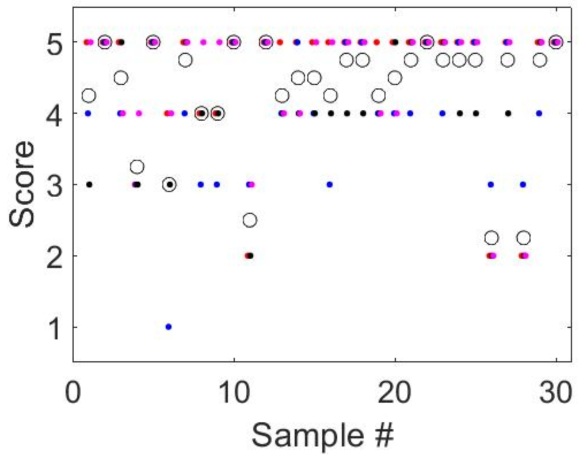


FIGURE 5.3: The results from a survey where four Tetra Pak® employees gave their opinion on the extracted colour and shadow lines from the validation data set. The filled dots correspond to the grading score given by an employee and the black circles mark the mean score for every sample.

5.4 Sample differences by machine settings

The impact of machine settings have been examined for the extended validation data set. The distance between the extracted colour lines and shadow lines for the samples can be seen in figure 5.4. For the results presented in this section the distance between the lines has been linearly interpolated to the common resolution of $m = 2000$ columns.

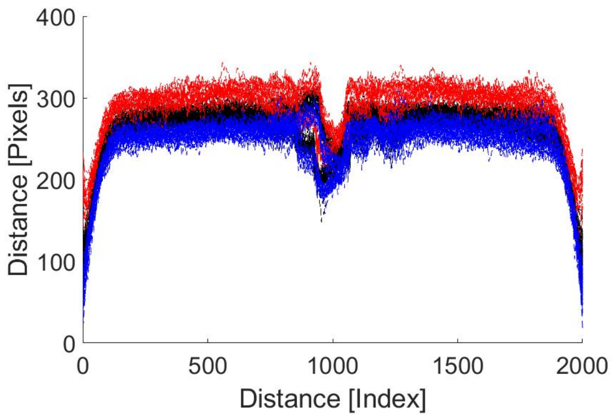


FIGURE 5.4: Distance between the extracted colour lines and shadow lines for the different machine settings. Low machine settings are shown in blue, medium in black and high in red.

The fraction of total variance explained by the 20 first principal components can be seen in figure 5.5. We choose to exclude all but the first three principal components, since almost all variance is explained by those.

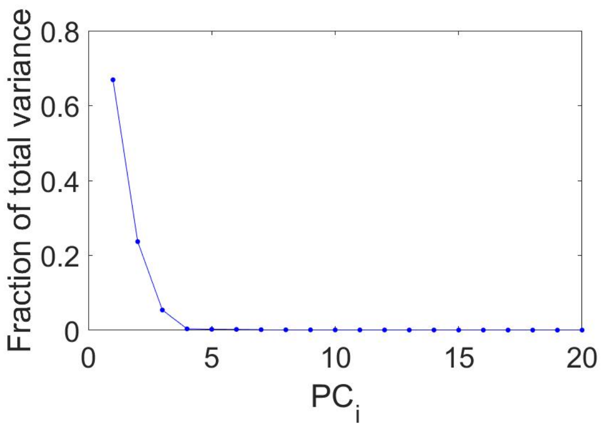


FIGURE 5.5: Fraction of total variance explained by the 20 first principal components.

The first three principal components can be seen in figure 5.6, where PC_1 is black, PC_2 is red and PC_3 is blue. Considering figures 5.5 and 5.4 we note that PC_1 explains the largest fraction of total variance and resembles the shape of the area between the lines. The second principal component is an almost horizontal line except for some structure close to either end. It most likely explains the difference in distance between the lines seen in figure 5.4, where higher machine settings yield larger distance between the colour and shadow lines. Finally the third principal component likely explains the two different paths seen at approximately index 900 in figure 5.4.

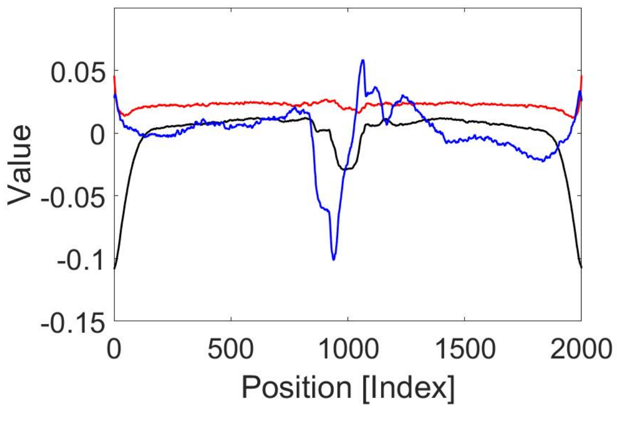


FIGURE 5.6: The first three principal components of the distance between the colour and shadow lines for the extended validation data set. Here PC_1 is black, PC_2 is red and PC_3 is blue.

A scatter plot of the principal component score for the three different machine settings can be seen in figure 5.7. Here low machine settings are shown in blue, medium in black and high in red. Note that there are two different clusters for every machine settings where the difference mainly is the score assigned to PC_3.

Finally the average reconstructed distance between the colour and shadow line for low, medium and high machine settings can be seen in figure 5.8. The reconstructions only use the first three principal components. The blue line corresponds

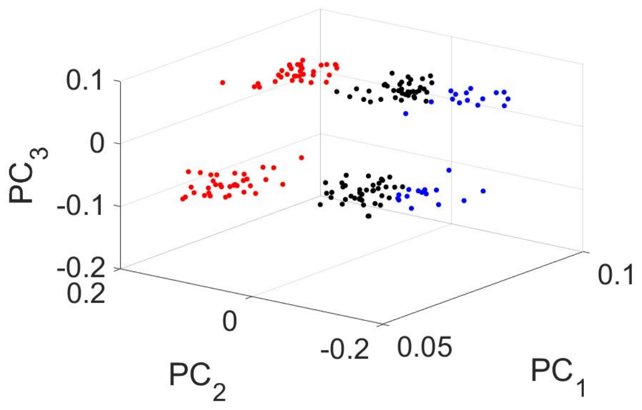


FIGURE 5.7: Scatter plot of the scores assigned to the first three principal components by images in the extended validation data set. Low machine settings are shown in blue, medium in black and high in red.

to low machine settings, the black line to medium and the red line to high.

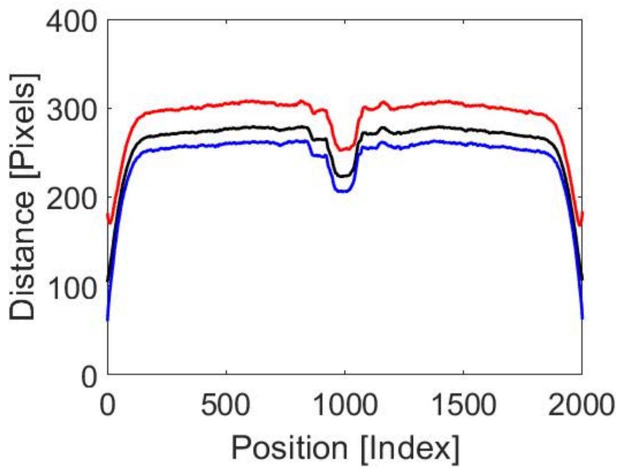


FIGURE 5.8: The average reconstructed lines for low, medium and high machine settings shown in blue, black and red. For this reconstruction only the first three principal components have been used.

6 Discussion

6.1 Thresholds and likelihood parameters

The calculated thresholds for the 30 images in the validation data set can be seen in figure 5.1 or with greater detail in table A.1. We note that the variation between samples is very small, which is to be expected given the manner in which they are calculated.

Since all samples are acquired using the same technique it is reasonable that the thresholds for different samples are similar. The variance is likely caused by differences in ambient light during the acquisition. The results are very satisfactory and this segmentation method is deemed to be effective.

The parameters for two likelihood images used to extract the shadow line can be seen in figure 5.2 or with greater detail in table A.2. The samples treated have great variance in both shape and colour of the shadow line, and hence variance in these parameters is to be expected.

Looking at the survey regarding the extracted lines in figure 5.3 we see that only five of the 30 samples are given a mean score lower than four. Taking a closer look at the samples with low score, namely samples $4, 6, 11, 26$ and 28, and comparing the likelihood parameters for those samples with the rest of the parameters in table A.2 no obvious pattern emerges. One could argue that sample 4 has an unusually large variance σ_{Mag}, or that sample 6 has expected gradient magnitude μ_{Mag} higher than every other sample and that this might explain the bad results, but it is hardly conclusive.

There are currently no conclusions to be made about whether the parameters for the likelihood images cause issues in the extracted lines or not. It is possible that a new and improved algorithm for estimation of these parameters would yield better results. Further work and comparisons between results for different algorithms would be needed to determine whether or not this would significantly improve the extracted lines.

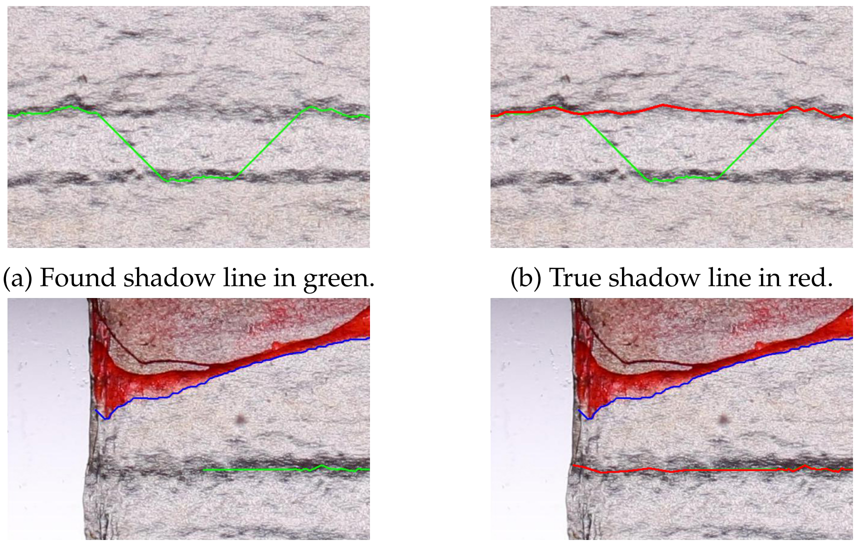
6.2 Extracted colour and shadow lines

The extracted lines are in many cases very good, but as we see in figure 5.3 or table A.3 there are some exceptions. First we note that the grades given by the four Tetra Pak® employees are similar, with the exception of sample 6. Despite the non linearity of the grading scale it is interesting to look at the mean grading score to get a feeling for how well the algorithms have performed.

Given that the definitions of what exact pixels constitute the two lines is a work in progress at Tetra Pak® we are satisfied with mean grading scores of four or higher. According to the scale this corresponds to there being no or minor errors in the extracted lines, and that these are likely to either not impact or have very small impact on further evaluation.

The extracted lines are relatively good, and only five of the 30 samples are given mean grading scores below four. While there is room for improvement these results are relatively satisfactory. To understand how the algorithms might need to be improved we take a close look at the errors in the five samples with mean grading scores below four, namely samples 4, 6, 11, 26 and 28.

For all five problematic samples there are issues with the shadow line. These come in two variations as seen in figure 6.1. The first variation is where short segments of the shadow line choose the wrong pixels following a dark line near the shadow line, seen as the green line in figure 6.1a. The second variation is when pixels are missing at either end, seen as the green line in figure 6.1c. Corresponding approximate manual lines have been added in red to figures 6.1b and 6.1d.


(a) Found shadow line in green.

(b) True shadow line in red.

(c) Found shadow line in green.

(d) True shadow line in red.

FIGURE 6.1: The worst examples of the two error variations which occurred in the shadow lines for the 30 model validation images. The found shadow lines are shown as green lines to the left and manually entered approximations of the true shadow lines as red lines to the right.

Further work on the algorithms would most likely benefit from error correction of the shadow line dealing with these issues. Issues such as the one in figure 6.1a could likely be detected by looking at the gradient of the extracted shadow line. The issue seen in figure 6.1c occurs because too many pixels are removed from the end of the line. This happens because the threshold T_{Rem} governing the removal of end pixels is badly chosen. Since this threshold is currently being manually chosen it should be easy to improve, for example by automatically choosing it through some process involving the pixels of the found shadow line.

6.3 Sample differences by machine settings

There are several interesting conclusions to be made from the PCA of samples acquired with different machine settings. Looking at figure 5.5 we note that the first three principal components explain almost all the variance in the data set. Further principal components are thus unnecessary and the samples are well described by their first three principal components.

Comparing the lines in figure 5.4 with PC_1 in figure 5.6 we notice great resemblance in shape. We further note that PC_1 is not able to describe the phenomenon at and around approximately index 900 on the x-axis in figure 5.4, where the lines appear to split into two different paths. This phenomenon is most likely described by PC_3 seen in figure 5.6.

Looking at the scatter plot in figure 5.7 we note that the principal component score given to PC_3 varies between positive and negative for samples of a given machine setting, essentially creating two clusters for every machine setting. This is likely caused by the two different paths previously described.

Further we notice that the score given to PC_2 varies greatly for the different machine settings. Looking at figure 5.6 we see that PC_2 is an almost horizontal line with some structure at either end, and the difference in score assigned to it therefore corresponds to differences in distance between the two sought lines depending on machine settings. This can also be seen in figure 5.4. In general higher machine settings increase the distance between the sought lines.

Using the first three principal components and calculating the average reconstruction of the samples in the extended validation data set yields the three lines in figure 5.8. We note that the general shape of the area between the colour line and the shadow line is similar for different machine settings. We further note that the distance between the lines is larger for higher machine settings, and that this effect is larger close to either end of the sample.

While there are differences in the distance between the lines for the three machine settings, especially at either end of the sample, these are not sufficient to explain the differences in the samples. The samples corresponding to a single machine setting can be further clustered into two groups each depending on the difference in principal component score for PC_3 in figure 5.7.

6.4 Conclusions

Algorithms which extract the colour and shadow lines in polymer samples have been developed. These are robust to changes in lighting, positioning of the sample as well as the shape and colour of the lines. The algorithms mostly yield good results but would likely be improved by adding error handling for the shadow line and changing how the ends of the shadow line are handled. The results might be further improved by changing how the first approximation of the shadow line is calculated.

Further we have seen that higher machine settings increase the distance between the colour line and the shadow line. The area between the lines mostly maintains its shape for all machine settings, but at either end of the sample high settings cause a disproportional increase in distance between the two lines, seen in figure 5.8. The machine settings are however not sufficient to describe all differences between the samples. Samples created with one machine setting can be further split into two groups depending on the score assigned to PC_3 seen in figure 5.7.

A Tables

A.1 Sample border thresholds

Sample #	T_{ls}	T_{ub}	T_{mix}
1	426.6826	25.7768	0.1874
2	428.1939	25.8607	0.1547
3	429.0205	26.3874	0.1724
4	429.5949	26.1503	0.1491
5	429.0588	26.0396	0.1499
6	429.5007	26.2211	0.1516
7	428.0382	26.1673	0.1850
8	429.6157	26.4783	0.1605
9	428.3433	25.9015	0.1662
10	429.2391	26.3180	0.1583
11	429.3169	25.6593	0.1594
12	427.6455	25.6629	0.1901
13	429.0849	25.8033	0.1597
14	428.9108	25.8257	0.1511
15	430.6974	25.6902	0.1510
16	429.1116	26.0512	0.1608
17	428.3972	25.3414	0.1571
18	429.2127	25.9106	0.1653
19	429.0948	25.4988	0.1729
20	429.2098	25.8831	0.1527
21	428.3534	28.0480	0.1773
22	429.3907	28.2987	0.1748
23	429.5654	25.6545	0.1598
24	428.8358	28.0266	0.1603
25	430.5335	25.9283	0.1560
26	428.8694	25.8273	0.1834
27	426.6207	25.7440	0.1921
28	424.2021	25.4789	0.2149
29	427.1930	25.1355	0.1749
30	425.9949	25.0403	0.1803

TABLE A.1: The threshold values automatically extracted from 30 input images in the validation data set. Three thresholds are calculated, T_{ls} above which the pixels match the light source, T_{ub} under which the pixels match the unilluminated background and finally T_{mix} above which the pixels of $I_{composite}$ match the red ink.

A.2 Shadow line likelihoods

Sample #	μ_{Int}	σ_{Int}	μ_{Mag}	σ_{Mag}
1	180.0127	35.3944	389.2537	107.3882
2	183.4188	30.359	425.5571	77.96329
3	137.9277	31.1182	472.5961	117.4897
4	174.5761	30.3966	471.6562	78.85767
5	167.6208	38.6512	444.4236	115.2314
6	153.6401	27.6053	526.6713	95.74503
7	171.9197	30.2198	422.5564	98.89746
8	170.5578	30.8961	498.6334	90.54493
9	166.693	30.735	447.8911	99.3143
10	174.015	30.2016	476.2979	83.52644
11	203.632	37.6627	355.0173	103.8492
12	188.95	38.4927	452.8094	108.5282
13	168.6398	28.184	448.9014	93.19084
14	226.4809	48.1318	343.9979	107.475
15	164.7838	31.1146	461.8016	98.42377
16	172.7406	37.2055	486.2061	103.2908
17	182.8176	32.895	404.0737	103.4984
18	187.6103	43.3658	464.9379	107.6889
19	157.3635	29.0122	458.8878	98.59724
20	226.5179	46.0651	357.5006	114.859
21	180.5758	34.5373	411.3655	99.19316
22	186.4965	39.2385	447.7426	95.69349
23	169.6217	32.9024	434.8302	97.09203
24	174.7718	35.5535	477.9945	90.05627
25	173.6933	31.4768	425.6738	104.6077
26	204.6026	40.3017	413.5659	105.7591
27	168.3804	35.0172	454.4919	109.7772
28	195.1314	40.5872	414.9701	96.18785
29	184.3838	35.1217	395.0898	103.4346
30	181.6086	38.1695	488.7637	98.85966

TABLE A.2: The expected values μ_{Int} and μ_{Mag} as well as the corresponding standard deviations σ_{Int} and σ_{Mag} resulting from the approximate lines calculated by the algorithm.

A.3 Survey: Quality of extracted lines

Sample #	Employee				Mean
	A	B	C	D	
1	5	4	3	5	4,25
2	5	5	5	5	5
3	5	4	5	4	4,5
4	3	3	3	4	3,25
5	5	5	5	5	5
6	4	1	3	4	3
7	5	4	5	5	4,75
8	4	3	4	5	4
9	4	3	4	5	4
10	5	5	5	5	5
11	2	3	2	3	2,5
12	5	5	5	5	5
13	5	4	4	4	4,25
14	5	5	4	4	4,5
15	5	4	4	5	4,5
16	5	3	4	5	4,25
17	5	5	4	5	4,75
18	5	5	4	5	4,75
19	5	4	4	4	4,25
20	5	4	5	4	4,5
21	5	4	5	5	4,75
22	5	5	5	5	5
23	5	4	5	5	4,75
24	5	5	4	5	4,75
25	5	5	4	5	4,75
26	2	3	2	2	2,25
27	5	5	4	5	4,75
28	2	3	2	2	2,25
29	5	4	5	5	4,75
30	5	5	5	5	5

TABLE A.3: The quality of the extracted lines as described by four Tetra Pak® employees.

Bibliography

Burt, P. and E. Adelson (1983). “The Laplacian Pyramid as a Compact Image Code”. In: 31.4, pp. 532–540. URL: `https://ieeexplore-ieee-org.ludwig.lub.lu.se/document/1095851`.

International Telecommunication Union (2011). *RECOMMENDATION ITU-R BT.601-7**. Tech. rep. International Telecommunication Union. URL: `https://www.itu.int/rec/R-REC-BT.601-7-201103-I/en`.

Jolliffe, I. T. (2002). *Principal Component Analysis*. 2nd ed.

Jähne, B. (2005). *Digital image processing*. 6th ed. Springer-Verlag Berlin Heidelberg.

Navarra, A. and V. Simoncini (2010). *A Guide to Empirical Orthogonal Functions for Climate Data Analysis*. Netherlands: Dordrecht : Springer.

Peters, J. F. (2017). *Foundations of Computer Vision. Computational Geometry, Visual Image Structures and Object Shape Detection*. Cham : Springer International Publishing : Imprint: Springer, 2017. ISBN: 9783319524832.

Sundararajan, D. (2017). *Digital Image Processing: A Signal Processing and Algorithmic Approach*. Springer Nature Singapore Pte Ltd. 2017.

Szeliski, R. (2011). *Computer Vision: Algorithms and Applications*. London: Springer London, 2011. ISBN: 9781848829350.

Image analysis as a tool for automatic evaluation and optimization

Today's industries are often dependent on automated processes. Image analysis can be an important tool to monitor such processes and certain features in the images are often used to quantify the quality.

While the important features in an image are often evident to the human eye, they are not generally trivial for a computer to find. Human vision can detect subtle changes of nuance, texture and colour. What information humans use to do this is not always evident or known, and thus it can be hard to get a computer to replicate the behaviour.

In general algorithms developed to detect certain features in images need to be robust to changes. The need for robustness varies depending on intended use, but might include for example the position, shape, colour or size of the features, as well as the lighting conditions. Occasionally the algorithms might need to detect features partially obstructed by foreground objects.

To find features in images several different techniques can be used and combined. Which techniques yield the best results is highly dependent on the evaluated images. An algorithm meant to detect two lines in images of polymer layers while being robust to changes in size, lighting and positioning has been developed and tested.

The results for the algorithm were good, and 25 out of 30 examined samples got satisfactory results. The quality of the extracted lines means that such an algorithm could be used in the industry given that it is supervised by a human operator. Whenever the algorithm produces an error the supervisor would need to manually correct it.

The algorithm could be further improved by adding so called error handling. Whenever an error occurs it should be automatically detected and the faulty pixels corrected. The resulting algorithm if such error handling was well implemented might be good enough for unsupervised evaluation.

To find one of the lines colour segmentation is used in conjunction with various morphological operations. The other line is found as the most likely line given by combined probability distributions considering pixel intensity, gradient magnitude, gradient direction and an approximate location.

The images of polymer layers evaluated depict polymer samples manufactured using different machine settings. The relative positions of the two lines for various machine settings were evaluated using principal component analysis.

It was found that the distance between the lines is dependent on the machine setting used as the sample was manufactured. Further it was found that there are differences in the relative positions of the two lines not explained by the machine settings, and to fully explain the process further classification would be needed. Since the machine settings used is correlated to the distance between the lines the optimal settings yielding a certain distance between the lines can be calculated.

Master's Theses in Mathematical Sciences 2019:E7
ISSN 1404-6342

LUTFMS-3363-2019

Mathematical Statistics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/