UNIVERSITY OF VAASA

FACULTY OF TECHNOLOGY

AUTOMATION ENGINEERING

Johnny Backlund

VISION SYSTEM FOR ONLINE DEFECT-DETECTION

An evaluation of methods for defect-detection for KWH Mirka's abrasives

Master's thesis for the degree of Master of Science in Technology submitted for inspection, Vaasa 26.1.2017

Supervisor	Prof. Jarmo Alander
Instructor	TkT Mats Sundell

FOREWORDS

During my second year at the university I discovered the field of machine vision. A project work that proved very challenging boosted my stubbornness, and my determination to succeed guided me through several books, papers and forums. Afterwards, machine vision algorithms remained as a personal interest.

In summer 2013 I was employed by a subcontractor to KWH Mirka. Among other things, they programmed machine vision systems. Here I would like to direct a special thanks to Tom Jansson who gave me a chance to familiarize with these systems. He also actively tried to find a thesis work for me and introduced me to people at Mirka.

One of these new contacts was Mikael Johnsson. Much thanks to him and Tom, I was employed by Mirka the next summer and got to work with camera and vision systems. This also led to this thesis work, which I have found really interesting. A huge thanks to Mikael for your invaluable support and help both before and during this work.

Further, I would also like to thank Professor Jarmo Alander for your help during all these years at the University, and Ph.D Mats Sundell at Mirka for your feedback and help during this work.

Finally, I would like to direct a big thank you to friends and family for your steadfast support.

Vaasa 26.1.2017

Johnny Backlund

TABLE OF CONTENTS

FOREWORDS 1

SYMBOLS AND ABBREVIATIONS 4

ABSTRACT 6

TIIVISTELMÄ 7

ABSTRAKT 8

1. INTRODUCTION 10
1.1. Application minimum requirements 12
1.2. Method 13
1.3. Related work 13

2. VISION SYSTEMS 15
2.1. Omron's vision systems 15
2.1. Machine vision lighting 17
2.1.1. Light sources 17
2.1.2. Lighting techniques 18
2.1.3. Lighting selection 21
2.2. Industrial cameras 22
2.2.1. Sensor types 22
2.2.2. Sensor size and resolution 23
2.2.3. Camera interfaces 24
2.2.4. Camera selection 25
2.3. Camera lenses 27
2.3.1. Field of view (FOV) 28
2.3.2. The f-number 29
2.3.3. Depth of focus and depth of field 30
2.3.4. Lens selection 32
2.4. Image processing methods 32
2.4.1. Morphological filters 33
2.4.2. Omron's Gravity and Area method 34
2.4.3. Omron's Defect method 35
2.4.4. The Fourier transform 35
2.4.5. Feature analysis with fuzzy membership functions 38

3. IMPLEMENTATION OF ONLINE MEASUREMENTS WITH THE FZ5-SYSTEM 42
3.1. Goals and requirements 42
3.2. Presentation of the FZ5 algorithm 45

4. OFFLINE DEVELOPMENT OF FREQUENCY SPACE APPROACH 49
4.1. Feature selection 50
4.2. Presentation of the algorithm 52

4.2.1. Parameter calculation 54
4.2.2. Defect detection 58

5. TESTING AND RESULTS 59
5.1. The FZ5-system 59
5.2. Texture analysis with the Fourier transform 67
5.3. Summary of the results 74

6. CONCLUSIONS 75
6.1. Future work 75

REFERENCES 77

APPENDIX A 79

APPENDIX B 81

APPENDIX C 87

APPENDIX D 92

SYMBOLS AND ABBREVIATIONS

f	Focal length
m	Magnification
CCD	Charge Coupled Device
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DMA	Direct Memory Access
DOF	Depth Of Field/Depth Of Focus
DSP	Digital Signal Processor
FCM	Fuzzy C-Means (clustering algorithm)
FFT	Fast Fourier Transform
FOV	Field Of View
FPGA	Field Programmable Gate Array
FZ5	Omron FZ5-L355 Vision System
GLCM	Grey Level Co-occurrance Matrix
GUI	Graphical User Interface
HMI	Human-Machine Interface

LCD	Liquid Crystal Display
LED	Light Emitting Diode
MB	Megabyte
MP	Megapixel
PLC	Programmable Logic Controller
SNR	Signal to Noise Ratio
ROI	Region Of Interest
USB	Universal Serial Bus
USB2	Universal Serial Bus 2.0
USB3	Universal Serial Bus 3.0

UNIVERSITY OF VAASA
Faculty of technology
Author: Johnny Backlund
Topic of the Thesis: Vision system for online defect-detection
Supervisor: Professor Jarmo Alander
Instructor: Ph.D Mats Sundell
Degree: Master of Science in Technology
Major of Subject: Automation Engineering
Year of Entering the University: 2009
Year of Completing the Thesis: 2017 **Pages:** 111

ABSTRACT

KWH Mirka has machines for punching out disks from their abrasive sheets. The sheets have gone through many processing stages during its fabrication, and as a result they contain different kinds of defects. Sheets have been joined several times with different techniques, the grip cloth has been joined, and test pieces have been carved out. These kinds of defects are made on purpose and are usually already marked, but not always. The abrasive roll may also contain unintended defects such as folds or stains.

The goal of this thesis is to develop a complete system, and also evaluate different methods, for detecting defects in Mirkas abrasive sheets. An Omron FZ5 vision system has been installed at the machine and the first goal is to configure this system so that it detects the most common defects. This system should perform according to the requirements set by KWH Mirka and the environment at the machine. This part of the work shall also give an idea of the suitability of the Omron-library for this task.

In an attempt to complete the texture processing shortcomings in the Omron-library, another algorithm based on features in frequency space will be tested with Matlab. Fuzzy membership functions are used to compare measured values against values from faultless material.

The result is a complete system for detecting the most common defects, along with an evaluation of some more advanced method's suitability for defect detection based on the pattern on abrasive sheets. This is all summarized to a recommendation on upgrades if more advanced measurements are to be done in the future.

KEYWORDS: Machine vision, defect detection, image processing, frequency space, quality control.

VAASAN YLIOPISTO
Teknillinen tiedekunta
Tekijä: Johnny Backlund
Diplomityön nimi: Konenäköjärjestelmä vikojen havaitsemiseen tuotantolinjalla
Valvojan nimi: Professori Jarmo Alander
Ohjaajan nimi: TkT. Mats Sundell
Tutkinto: Diplomi- insinööri
Oppiaine: Automaatiotekniikka
Opintojen aloitusvuosi: 2009
Diplomityön valmistumisvuosi: 2017 **Sivumäärä:** 111

TIIVISTELMÄ

KWH Mirkalla on lävistyskoneita, joilla stanssataan pyöröjä hiontamateriaaleista. Materiaalissa on lukuisia erilaisia vikoja jotka ovat syntyneet hiontamateriaalirullan valmistuksen eri vaiheissa. Materiaalissa on erilaisia liitoksia, tarrakangasta on jatkettu ja siitä on leikattu testikappaleita. Nämä viat ovat tietoisesti tehty ja ovat usein valmiiksi merkitty, mutta eivät aina. Myös tahattomia vikoja, kuten taitoksia ja tahroja, voi esiintyä.

Tämän diplomityön tarkoituksena on kehittää toimiva järjestelmä, ja arvioida erilaisia vikojen havaitsemismenetelmiä. Koneeseen on asennettu Omronin FZ5-konenäköjärjestelmä, ja ensimmäinen tavoite on tämän järjestelmän konfigurointi löytämään tavallisimmat viat. Tämän järjestelmän pitää täyttää KWH Mirkan sekä työympäristön asettamat vaatimukset. Tämän työvaiheen tarkoitus on myös antaa käsitys Omron-kirjaston sopivuudesta tehtävään.

Yrityksessä täydentää puutteet Omron-kirjastossa arvioidaan myös toinen metodi, joka analysoi kuvan taajuusspektriä Matlabilla. Sumeiden jäsenyysfunktioiden avulla vertaillaan mitattuja arvoja ja arvoja virheettömien tuotteiden koekuvista.

Työn tuloksena on järjestelmä joka havaitsee tavallisimmat viat ja arviointi taajuusspektrimenetelmän sopivuudesta tunnistaa vikoja hiomamateriaalin kuvioinnin perusteella. Saadut tulokset on koottu suositukseksi päivityksiin jos tulevaisuudessa halutaan suorittaa vaativampia mittauksia.

AVAINSANAT: Konenäkö, laaduntarkistus, kuvankäsittely, taajuusavaruus, laadun mittaus.

VASA UNIVERSITET
Tekniska fakulteten

Författare:	Johnny Backlund	
Titel:	Vision system för online defektdetektering	
Övervakare:	Professor Jarmo Alander	
Handledare:	Ph.D Mats Sundell	
Examen:	Diplomingenjör	
Inriktning:	Automationsteknik	
Studiernas inledelseår:	2009	
Avhandlingens färdigställande:	2017	**Antal sidor:** 111

ABSTRAKT

KWH Mirka har maskiner som stansar ut sliprondeller ur en rulle med slipmaterial. Detta material har genomgått flera olika steg under tillverkningen och som resultat av detta så innehåller det olika defekter. Materialet har blivit skarvat med olika tekniker, grip-tyget på baksidan kan innehålla skarvar och man har skurit ut provbitar. Dessa defekter genereras medvetet under tillverkningen och är ofta färdigt uppmärkta, men inte alltid. Också oväntade defekter så som veck och fläckar kan förekomma.

Denna avhandling går ut på att utveckla ett komplett system, samt utvärdera olika metoder, för att upptäcka defekter i Mirkas slipmaterial. Ett Omron FZ5 system har blivit installerat på maskinen och det första målet är att konfigurera detta system så att det upptäcker de vanligaste defekterna. Detta system skall uppfylla de krav som ställs av KWH Mirka samt av omgivningen vid maskinen. Vidare så skall denna del av arbetet leda till en uppfattning av Omron-biblotekets lämplighet för ändamålet.

I ett försök att komplettera begränsningarna i Omron-biblioteket så testas också en annan metod. En algoritm baserad på värden från bildens frekvensspektrum kommer att testas i Matlab. Oskarpa medlemskapsfunktioner (*fuzzy membership functions*) kommer att användas för att jämföra uppmätta värden med värden från exempelbilder av felfritt material.

Resultatet är ett komplett system som skall upptäcka de vanligaste defekterna, samt en utvärdering av några mera avancerade metoders lämplighet för defekt igenkänning i mönstret på slipmaterial. Detta sammanställs till en rekommendation på uppgraderingar om man i framtiden vill utföra mera avancerade mätningar.

Nyckelord: Maskinsyn, defekt detektering, bildbehandling, frekvensrymd, kvalitetskontroll.

1. INTRODUCTION

Machine vision algorithms usually demand quite a lot of computing power. Along with the development of faster and cheaper computers, vision systems are finding their way into more and more demanding industry applications. Vision systems are frequently used in quality control tasks, such as identifying parts, defect detection, sorting, code scanning, positioning and dimension measurements.

To build a complete machine vision system, there is a lot more to consider than simply the algorithms needed. Sufficient lighting, fast enough hardware, and a software library that covers your needs are all key components to create a robust solution. Fast and stable system performance is achieved by careful selection of each of these components. This thesis will include some basic theory on the key components of vision systems.

The first goal for this thesis is to use machine vision to detect the most common defects in KWH Mirka's abrasives. At the machine that punches the disks for sanding machines, defects need to be detected so that only first quality material is packed and sent to the customers. If there is a problem with the grip cloth on the backside of the material, this might also cause the cloth to get stuck in the machine when punching. This means that the system needs to process the images fast enough so that the defects do not reach the punching stage before the machine is stopped. Some defects are shown in Figure 1.

An Omron FZ5-system has been installed at the machine prior to this work. These systems are used in several other applications at Mirka and are therefore familiar to the developers. Due to economical reasons and low minimum requirements, a quite slow version was chosen. My task is to configure the system so that it meets the requirements described in the next section. Also, by testing several different methods, the capability of the software library and this version of the FZ5-system will be evaluated. This will give a hint on what upgrades would be needed to perform more advanced measurements.

Further, in an attempt to overcome the limitations of the Omron-library, and to provide more ideas on upgrades, another approach is also tried. A second algorithm is evaluated in Matlab, which uses the Fourier-transform and extracts a number of features from the image's frequency space. The idea is that changes in the repetitive prints on the back-side of the material should alter the frequency spectrum. The features are evaluated with Gaussian membership functions that will give a total membership value indicating how similar the image is to faultless material. This algorithm will only be tested offline and will not be taken into use at this stage.

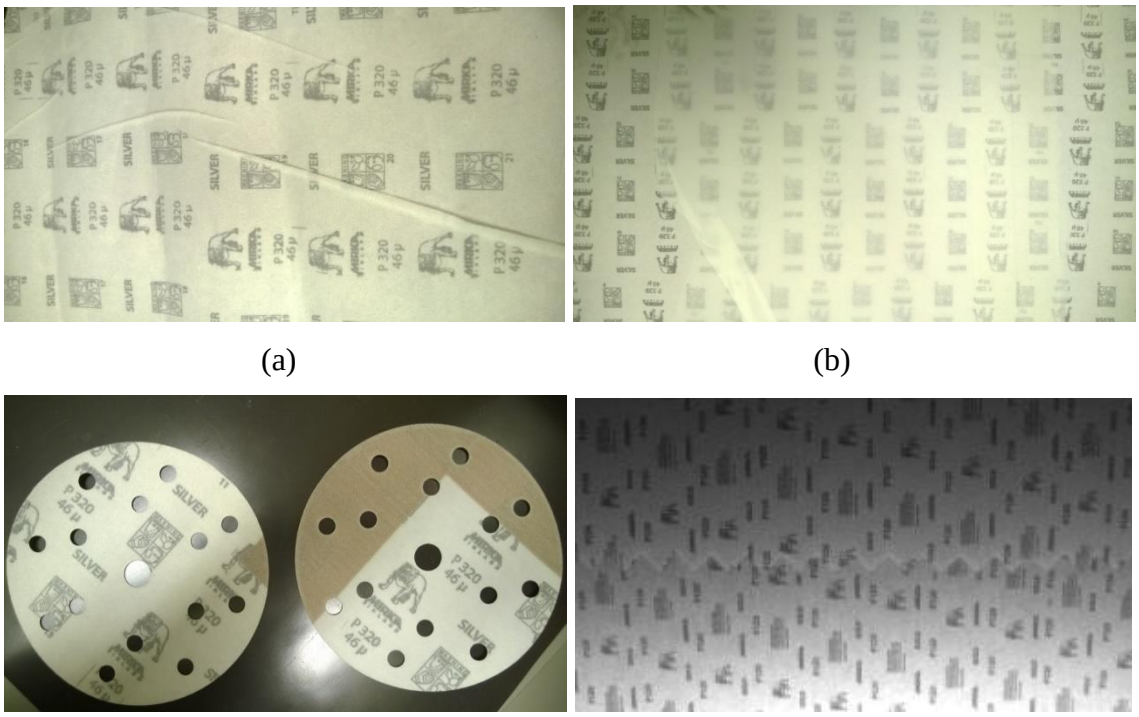


Figure 1. Examples of defects. (a) Folds in the grip cloth. (b) A section where the grip cloth is overlapping. (c) A couple of disks that have been punched from a tape joint that slipped through. (d) A joint made by the "maker" i.e. the machine that produced the roll of abrasive.

1.1. Application minimum requirements

The speed of the material can be up to about $v \approx 85$ m/min ≈ 1.5 m/s. The distance seen by the camera is about 0.5 m. The time interval between consecutive measurements must therefore be shorter than $t_{dmax} \approx 0.5$ m / 1.5 m/s = 0.333 s. The punching stage is located about 2.0 m after the measurement area. The machine must come to a complete standstill within $t_{stop} \approx 2.0$ m / 1.5 m/s = 1.33 s after a defect is detected.

Two cameras will be used to monitor the material. The width of the material is about 1500 mm. The field of view for each camera should therefore be at least 750 mm broad. A sketch of the setup is shown in Figure 2.

The system should not require any actions by the operators other than when a defect is detected. The system should therefore start and stop when the machine is started or stopped and it should detect which material is loaded into the machine by itself, and adjust its parameters accordingly. The operators must also be able to disable the system from the machine HMI if they need to.

Robustness should be prioritised instead of sensitivity, i.e. machine stops because of incorrect measurements should be minimized, even though it might affect the system's ability to detect some defects. Already a few unnecessary stops will probably result in irritated operators that keep the system disabled all the time.

The system must reliably detect at least tape joints and holes in the material.

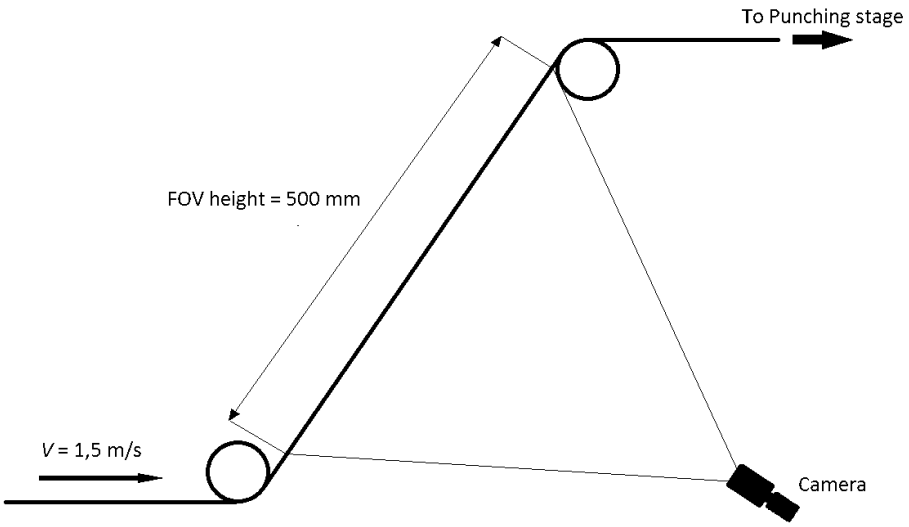


Figure 2. Sketch of the camera setup for the online FZ-system.

1.2. Method

The Omron system was used to log images at the machine. By analysing and performing tests on the logged images, the algorithms and their parameters will be decided upon. The majority of the tests will be performed using Matlab. Simulator software for the Omron system that can be installed on a PC is also available and can be used to experiment without having the camera system connected. The study of related work will also help in developing the algorithms.

1.3. Related work

Defect detection is a broad term that includes lots of applications. In most cases, the task is to detect defects on discrete parts, which involves methods not applicable in this case. Quite a lot of research has been done however, to detect defects on fabric and printings. These applications can be of interest since the setup is similar; defects are to be detected from a material that continuously flows past a camera.

Chan and Pang (Chan & Pang, 2000) used Fourier analysis to develop a method for defect detection. They use seven features extracted from the frequency magnitude spectrums in the x- and y-directions to describe faultless material. The method is translation invariant and well suited to detect changes in the repetitive patterns of the yarns in fabric. Their work is described more precisely in Chapter 4, since the algorithm developed in that chapter is also based on features in the Fourier magnitude spectrum.

Aziz *et al*. (Aziz, Haggag & Sayed, 2013) have developed an algorithm based on morphological operations and the discrete cosine transform (DCT) to detect defects in uniformly coloured fabric. They use morphology to highlight intensity variations in a greyscale image. With the use of the DCT and Otsu's method, they then find a suitable threshold to separate the defects from the background. Their method does not rely on models of faultless material, but it is impractical when dealing with patterned material.

Wavelets have also been widely used in defect detection. Compared to the DFT, the wavelet-transform preserves time (position) information and enables the system to locate the defect. Yang *et al*. (Yang, Pang & Yung, 2005) have developed a method for designing wavelets for detecting specific defects, called adaptive wavelets. In their work, they design a wavelet for each defect that is to be detected. The use of several wavelets results in better sensitivity than with a single wavelet transform, and it enables the algorithm to more reliably classify the defect. After the wavelet transforms are complete, only the wavelet that gives the highest error measure is used in further analysis. This algorithm would probably require parallel processing of all the wavelet transforms to be fast enough for online measurements.

Zuo *et al*. (Zuo, Wang, Yang & Wang, 2012) uses a method where they first enhance the texture of the fabric by using a non-local means filter algorithm. This algorithm uses all pixels in an image, weighted by a "similarity measure", to calculate the new value for an individual pixel. After the enhancement, grey level co-occurrence matrices (GLCM) are used to extract features from the image. GLCMs for four different directions are calculated. The features extracted from each GLCM are the angular second moment, the contrast, the correlation and the entropy, which results in a feature vector of total length 16. An Euclidian distance classifier is used to classify the vectors.

2. VISION SYSTEMS

To build an online vision based measurement system, is usually more challenging than performing offline research, since a lot of environmental conditions needs to be considered. Might dirt or dust cause problems? Is there a need to block out ambient light with enclosures? Or maybe the application lighting should be enclosed due to ergonomic aspects? It is usually preferable to start with selecting the hardware, such as lighting, cameras, and lenses. Changing any of these things later will probably result in changes in the measurements, and a need for changes in the algorithm. That is why design is so important. User friendliness is also of importance when implementing online systems.

This chapter starts with a presentation of Omron's vision systems, especially the FZ5-system that is used in this work. The following sections contain theory on lighting, industrial cameras and optics. At the end of each of these sections, the hardware used with the FZ5 is described along with a discussion of the relevant properties.

After the hardware sections follows descriptions of the software methods used in this work.

2.1. Omron's vision systems

Omron has a range of machine vision systems and measurement sensors for vision applications. Their range of vision sensors include, among others, the FH-series high speed vision systems and the lighter FZ5-series. The FH-controllers utilizes multicore CPU's and parallel FPGA based frame grabber circuits that can handle up to 8 connected cameras. (Omron, *Quality Control & Inspection Guide 2014*)

The FZ5-series is specially designed for positioning and inspection tasks and come in several models, classified as "high speed-", "standard-" or "lite-" controllers. The system used in this work is a FZ5-L355, and this is the system referred to as "the" FZ5-system. This model is a "lite" controller that supports two cameras. It is equipped with standard VGA output port for connection to a separate monitor. Interfaces for commu-

nication include a RS-232 port, Ethernet, two USB 2.0 ports, and a parallel communications port. The parallel port will be used to communicate with the machine PLC. Three different camera resolutions are supported: 0.3 MP, 2 MP and 5 MP. Only Omron's own cameras and interfaces can be used with the system. As mentioned in the introduction, the Omron system was chosen mainly because it was familiar to developers at Mirka and also due to economical reasons. The minimum requirements, that the system should detect holes and tape joints, are quite easy to achieve and do not require costly high-end equipment. One goal for this work is indeed to give recommendations on upgrades if better performance is desired in the future. (Omron, *Quality Control & Inspection Guide 2014*)

The complete software library consists of seven different groups. The "Input Image" group contains blocks for controlling cameras. The "Compensate Image" group contains pre-processing tools such as image transformations and filters. Inspection functions are located in the "Measurement" group. Here one finds search functions for different shapes and templates, corner and intersection detectors, classification functions, colour data and area calculations, defect detection tools, edge measurements and character and barcode readers. The remaining four groups contain methods for altering program flow, data handling and arithmetic operations, output and communications, and displaying results. Macros can also be used to implement own simple functions. (Omron, *Vision Sensor-FH/FZ5 Series Vision System Processing Item Function Reference Manual)*

The system has reserved memory for image logging, but it is quite limited. The number of images that can be saved depends on the number of cameras and their resolution. With two 2 MP-cameras, 20 images can be saved. It is however, possible to connect an external memory device to an USB port to save logged images to. (Omron, *Quality Control & Inspection Guide 2014*)

2.1. Machine vision lighting

A camera measures light intensity. The appearance of an object, and thus the measurement result, therefore depends highly on the lighting conditions. By varying intensity, angles, and wavelengths of the light, one can bring out the preferred details. Correct lighting can greatly simplify a problem, while inadequate lighting can make a simple problem almost impossible. The lighting is something that often is not considered until the end of the project, although it is one of the components that have the greatest impact on system performance. Two things that can play a central role when designing the lighting are the light sources spectral content, and the lighting technique. These things will be discussed in the following two sections. (Microscan 2013)

2.1.1. Light sources

Different types of light sources have different benefits. The most common light sources in use today are fluorescent, quartz-halogen and LED sources. When higher brightness is needed also xenon, metal halide and high pressure sodium sources are in common use. Lasers are often used to get a structured light pattern that can be used to detect topography details. (Martin 2007)

LEDs have developed over the last years to become a suitable light source for many applications (Microscan 2014). The advantages of LEDs have in fact resulted in that many manufacturers of machine vision lighting do not have other sources at all in their basic product line. They consume small amounts of power, their lifetime is long, they are small and flexible, and also cost effective. LEDs do degrade during their lifetimes, but not to the same extent as many other light sources. LEDs can be strobed at arbitrary frequencies and duty cycles, and strobing lengthens their lifetime. Only a few years ago, LEDs fell short when it came to light intensity, but as of 2014 LEDs outperform many other sources also in this respect (Vision light tech 2014)(Pinter). The light intensity is directly proportional to the forward current through the LED, and as long as the thermal limits aren’t violated the current can be freely adjusted (Pinter). In true-colour imaging their spectral content is not optimal, but RGB and white LEDs are improving all the time. (Martin 2007)

Monochrome light is also available with LEDs, and can be utilized in several ways. It can for example be combined with filters to block out ambient light. The use of monochrome light usually also result in better image resolution since chromatic aberrations in lens systems are eliminated. Further, longer wavelengths penetrate bio and plastic materials better and can be used to see through thin and opaque pieces, while shorter wavelengths are better to bring out surface details. One can also match the wavelength of the light source to the colour of objects that one wants to bring out in an image. This is shown in Figure 3. (Martin 2007)

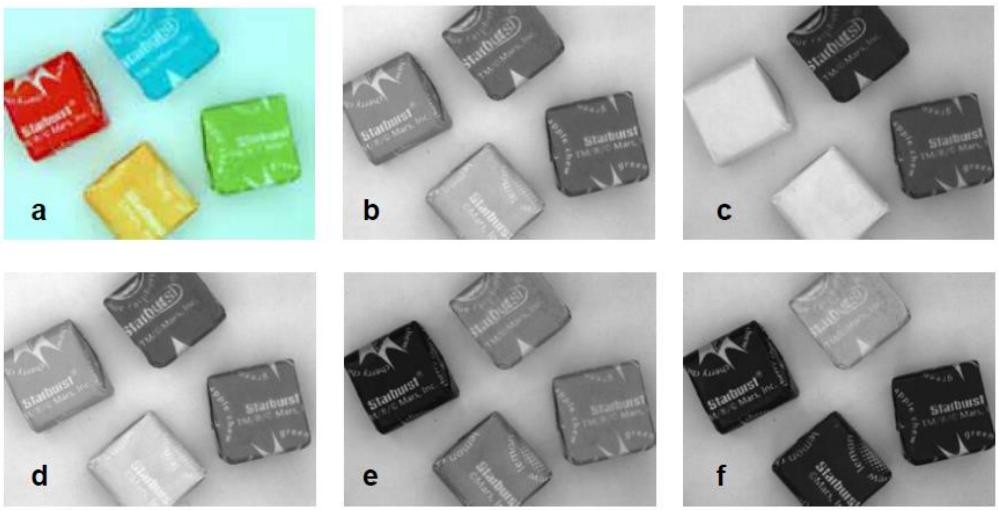


Figure 3. The effect of different wavelengths on pieces of candy. Image a – white light and color CCD camera. Image b – white light and B&W camera. Image c – red light. Image d – red and green light (yielding yellow). Image e – green light. Image f – blue light. (Picture: Martin 2007)

2.1.2. Lighting techniques

Knowledge about the geometry of inspected objects and about the reflection of light is another key to successful lighting. Different techniques can be utilized to generate high contrast on the right features.

Backlight is perfect for bringing out object contours or holes. The light source is placed directly behind the object and forms a bright background. The object blocks the light and appears dark. The method generates high contrast images, and is often used to de-

tect objects, specify orientation, or measure dimensions. The principle is shown in Figure 4. (Martin 2007)

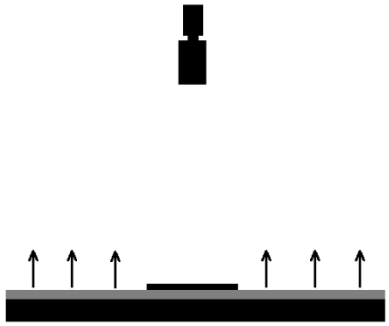


Figure 4. Backlight.

In **bright field lighting**, the light source is placed so that light is reflected of surfaces into the camera. Rough surfaces or missing material scatters the light in all directions and appear darker. (Microscan 2013)

An example is shown in Figure 5. This technique is called “partial bright field lighting” or “directional lighting”, since the inspection object is lit from one side only. Surface irregularities thus cast shadows, and become visible.

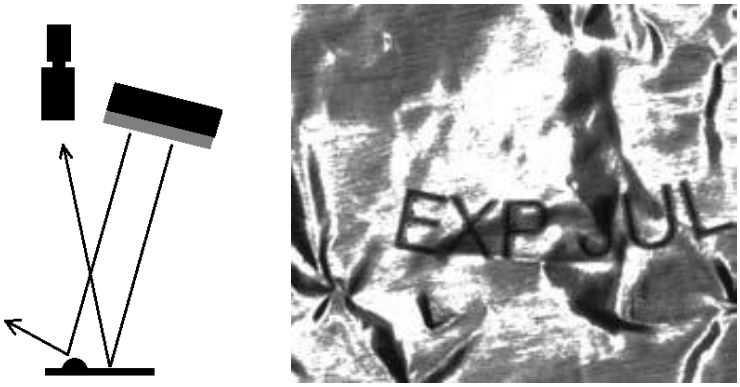


Figure 5. Partial bright field lighting. Because of the directional nature, the shape of the surface appears when irregularities form shadows, but the contrast of the text is bad. (Picture: Microscan 2013)

If the object is lit equally from each side, one speaks of “full field” lighting. This can easily be achieved by the use of ring lights. Another quite common way to achieve this is the use of a “diffuse dome”. The light is reflected from all directions onto the object and thus eliminates shadows. This is shown in Figure 6.

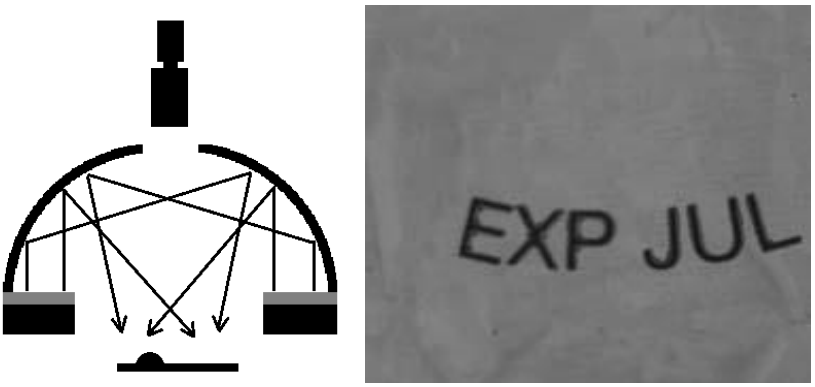


Figure 6. Full bright field lighting with diffuse dome. Shadows are eliminated so the shape of the surface is no longer visible, but the uniform lighting produces high contrast text. (Picture: Microscan 2013)

Another method that is classified as full bright field lighting is the use of on-axis diffuse lighting. A beam splitter is used to align the light with the optical axis of the camera. As soon as the surface is not perpendicular to the camera, the light is reflected away from the axis and does not return to the camera. An example is shown in Figure 7.

Figure 7. Full bright field lighting using on-axis diffuse lightning. Non-flat areas appear as dark spots. (Picture: photonics.com)

While in bright field lighting, flat surfaces reflect the light back to the camera and thus appear bright, **dark field lighting** results in the opposite. The light source is placed in such an angle that the light is not reflected into the camera by a flat surface. Instead, imperfections such as scratches or rough surfaces that scatter the light appear brighter to the camera. Dark field lighting can also be implemented as partial- or full field. An example of partial dark field lighting is shown in Figure 8.

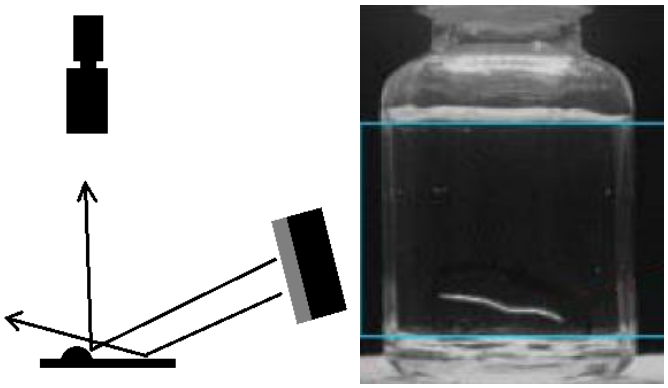


Figure 8. Partial dark field lighting. The glass bottle is lit from above, and the glass does not scatter the light very much so it appears dark. The crack on the other hand becomes visible. (Picture: Microscan 2013)

2.1.3. Lighting selection

In the case of defect detection on abrasives, we are to inspect a flat surface with prints on it. This implies that we should use bright field lighting. Since we are not searching for topographic details, full bright field lighting with completely uniform lighting over the whole area would be the ideal solution from a functional point of view.

A diffuse dome is usually used in challenging setups where uniform lighting is needed on uneven or highly reflective surfaces. In this case the surface has neither of these properties, and the use of domes would simply be an unnecessarily expensive solution. The area that should be inspected for defects is also quite large, so several domes would be needed.

A cheaper solution is to use line lights in a partial bright field configuration. Placing one light above, and one under the cameras gives us quite uniform lighting in the middle of the inspection area. The longer the line lights, the better the approximation of a full field setup.

This application do not require special attention to spectral content, so white light is used to provide possibilities for colour processing if needed in the future. The measurement area is also open and visible, so white light is also preferred from an ergonomic point of view.

The abrasive roll is about 1500 mm broad. 1700 mm line lights are made of LED strips and placed under and above the cameras. The line lights are thus 200 mm broader than the abrasive roll and have a 100 mm “overlap” on both sides. This overlap is actually a bit too small. We do not get uniform lighting over the whole material which is darker at the edges than in the middle. The lights could not easily be made longer due to limited space. Using mirrors or additional light sources at the edges could further improve the lighting, but it was decided that the current setup was good enough for this application.

2.2. Industrial cameras

A camera is a component with many specifications to consider, and manufacturers may have hundreds of different models in their range. An industrial camera consists of an image sensor, data processing circuits and an interface that uses a specific protocol to communicate with a computer. In the following sections, a number of camera properties and parameters will be explained and discussed.

2.2.1. Sensor types

There are two commonly used sensor types, CCD and CMOS. Historically CCDs have been used in professional high quality imaging and CMOS in cheaper consumer electronics. As of 2014, CMOS technology has however developed to become the sensor of

choice in the majority of applications. (Rashidian & Fox, 2011)(Schwär & Toth, 2014) (Niederjohann 2014)

A CMOS sensor pixel consists of both the light sensitive element and conversion electronics. The electronics converts the charge built up by the pixel directly into a voltage that can be read out in a parallel fashion. Other circuits can also be implemented on the same chip, resulting in fast and small sized cameras. CMOS sensors have low power consumption, high resolution, and an attractive price/performance ratio. The parallel reading of pixels also enables windowing, i.e. only part of the pixel values are read out, and can thus further increase speed.

Additional electronics come at a price however. This is because the light sensitive components only covers part of the sensor surface, and results in lower light sensitivity. A lower sensitivity to incoming light means lower total power of the pixel output signals, and leads to a lower signal to noise ratio (SNR). This might cause problems when imaging in light starved environments, such as astronomy. The electronic circuits themself also induce more noise, resulting in even lower SNR. However, today's high end CMOS sensors have been developed to such a degree that their SNR levels are comparable to CCD's, and they outperform CCD's in many other aspects. (Rashidian & Fox, 2011)(Schwär & Toth, 2014) (Niederjohann 2014)(Koljonen 2013)(Litwiller 2005)

In a CCD sensor, the light sensitive elements cover the entire surface of the sensor, giving it a high sensitivity to light compared to a CMOS sensor of the same size. When reading the image, the charges built up by the pixels are shifted out to a common node where the voltage conversion takes place. The conversion into a digital signal and further processing is done by external circuits. CCD sensors delivers high image quality in slower applications, and are better suited for imaging in environments with sparse lighting. (Schwär & Toth, 2014) (Niederjohann 2014)(Koljonen 2013)(Litwiller 2005)

2.2.2. Sensor size and resolution

The resolution of an imaging system is a measure of the minimum distance needed between two real points, so that they can be perceived as separate in the image. The reso-

lution of a digital camera is given as the number of pixels on the image sensor. A higher resolution camera has the potential ability to detect smaller details from the same scene, but it requires a lens that is able to project such small details. The camera resolution should not be higher than required for the application due to speed considerations. Transmitting larger amounts of data takes longer, and if the image is to be processed by a vision algorithm, the number of pixels might have a huge impact on time needed. (Schwär & Toth, 2014)

As an alternative to the area scan cameras commonly used, there are also line scan cameras. The sensors of these cameras have their pixels arranged in only 1, 2 or sometimes 3 rows. These cameras can deliver high resolution images with high frame rates. Line scan cameras are usually used in high speed applications where the objects or material move past the camera. An image can then be built up in software from sequential frames captured. High frame rates require short exposure times, which in turn require intense lighting. (Schwär & Toth, 2014) (Koljonen 2013)

2.2.3. Camera interfaces

The interface between the camera and the processing system is one of the most important to consider, because they differ significantly from each other. The most common interfaces today are GigE Vision, USB3 Vision, and Camera Link. Older interfaces also include USB2 and Firewire, but these are inferior to the newer ones and are not recommended for use in new setups. (von Fintel 2013)

The Gigabit Ethernet interface is the slowest of the three main interfaces even though its' bandwidth of about 100 MB/s is enough for many applications. Due to other properties, it is still the fastest growing interface based on the number of installations. The main advantage is the possible cable length of up to 100 meters without amplifiers. Ethernet ports are standard hardware on any PC system which enables easy integration, and it is also easy to configure the interface for multiple cameras. The Gigabit Ethernet interface loads the CPU of the processing system since copy processing is required to move data. (von Fintel 2013)

USB3 is an interface that is also flexible since USB3 ports are also common hardware on newer systems. With a maximum bandwidth of 350 MB/s it is significantly faster than Gigabit Ethernet. The use of DMA circuits also reduces CPU load to a minimum. The interface is also suitable for battery powered systems due to low power consumption and suspend modes. Disadvantages include the short cable lengths of up to 8 meters and complex setups for multiple cameras. This interface is commonly used to replace older USB2 or Firewire setups, and is usually also preferred instead of the slower versions of the Camera Link interface due to cost considerations. (von Fintel 2013)

Camera Link is an interface especially designed for vision applications. Three versions are available: base, medium and full. The base and medium versions has bandwidths of 255 MB/s and 510 MB/s respectively, while Camera Link full can achieve up to 850 MB/s. Camera Link requires a special frame grabber card between each camera and the processing system. Frame grabbers, cables and connectors must be certified by the manufacturer and compatible with other components in the setup. This makes the whole system complex and expensive compared to the other interface solutions. Due to this, USB3 is usually preferred instead of Camera Link base and medium. Camera Link support cable lengths of about 10 meters. (von Fintel 2013)

2.2.4. Camera selection

As mentioned in the introduction, the FZ5-system had been installed at the machine already before my work was started. So were also the two cameras, both with a resolution of 2 MP. This is unnecessarily high for this application, and a lower resolution would have sped up both the image transmission and processing.

Only Omron's own cameras can be used with the FZ5-system, and their range of models is quite limited. All models have one of three resolutions: 0.3 MP, 2 MP or 5 MP. (Omron, *Quality Control & Inspection Guide 2014*)

All 0.3 MP models have an image format of 640x480 pixels. A horizontal FOV of 750 mm would have meant that the width projected onto one pixel w_p = 750 mm/640 pixels = 1.17 mm. According to the Nyquist sampling theorem, it is theoretically enough to

detect details of sizes larger than $2w_p$ = 2.34mm. This would have been enough for this application.

All cameras for the FZ5-system are equipped with CCD-sensors. High-speed CMOS sensors are available only for the FH-series vision systems. Two series based on physical size are available. The specifications for the standard series of cameras for the FZ5-system are shown in Table 1. There are also a series of "small CCD cameras", which all have a resolution of 0.3MP, and intelligent cameras. The interface for all models is based on Camera Link (Omron, *Vision sensors FH/FZ5 Vision Systems User's Manual*). (Omron, *Quality Control & Inspection Guide 2014*)

FZ-S2M cameras are used in this work, but as can be seen from Table 1, the FZ-S camera has more than twice the speed of the FZ-S2M, and the total number of pixels to analyse would have been only about a sixth of the pixels of the FZ-S2M. Colour images are not needed and therefore monochrome cameras are used.

Table 1. Specifications for Omron's standard series cameras for the FZ-system.

Model name	**FZ-S**	**FZ-SC**	**FZ-S2M**	**FZ-SC2M**	**FZ-S5M**	**FZ-SC5M**
Sensor type	Monochrome	Colour	Monochrome	Colour	Monochrome	Colour
Sensor size	1/3-inch	1/3-inch	1/1.8-inch	1/1.8-inch	2/3-inch	2/3-inch
Resolution	640x480	640x480	1600x1200	1600x1200	2448x2044	2448x2044
Pixel size	7.4 µm	7.4 µm	4.4 µm	4.4 µm	3.45 µm	3.45 µm
Frame rate	80 fps	80 fps	30 fps	30 fps	16 fps	16 fps
Image read time	12.5 ms	12.5 ms	33.3 ms	33.3 ms	62.5 ms	62.5 ms

If cameras could have been chosen without limiting ourselves to Omron's cameras, a CMOS sensor would have been the natural choice, since the quality of the CCD is not needed and a CMOS sensor is more flexible when it comes to reading pixels. It could also have been an idea to consider using a line scan camera. Area scan cameras are easier to start using, since a line scan camera requires precise synchronisation with the

speed of the material and the image reconstruction requires setting extra parameters. The speed of an area scan camera is also enough in this case, it is the processing of the images that takes up the majority of time. The benefit for using line scan cameras in this case would be the simplification of the lighting setup. Getting uniform lighting over an inspection area this large is difficult, while it would be easier to obtain it over a single line.

2.3. Camera lenses

Most industrial cameras do not come with fixed optics, but are instead compatible with standard lenses that can be bought separately. The ability to freely choose the lens is economically beneficial, since high quality optics is expensive.

No real lens is ideal. They suffer from different distortions, and usually “the lens” actually is a system of several lenses, which together tries to approximate an ideal lens as well as possible. A lens has a certain resolution which defines how small details it is able to project without blurring them out. In vision applications where resolution is critical, for example when measuring distances and details with small tolerances, a high resolution lens is of highest importance.

The thin lens equation models an ideal lens and is written:

$$\frac{1}{a} + \frac{1}{b} = \frac{1}{f} \tag{1}$$

where a is the object distance from the lens, b is the image distance from the lens and f is the focal length of the lens (Figure 9.)

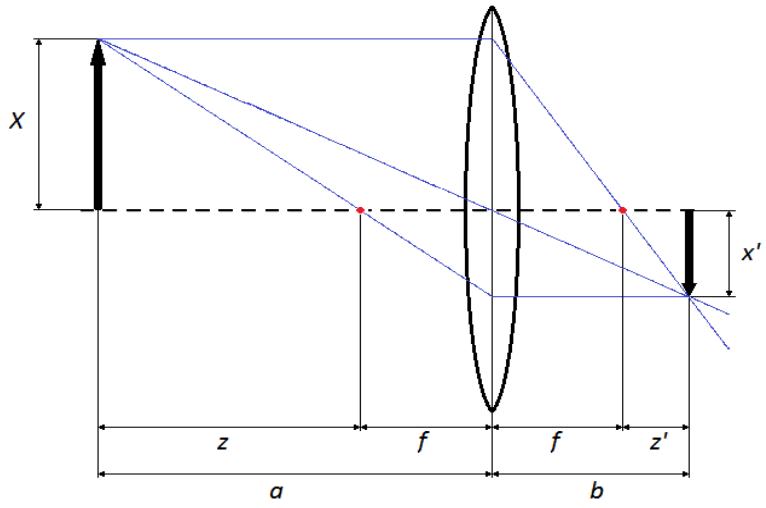


Figure 9. The thin lens equation. *X* and *x'* are the object and image heights respectively, *a* and *b* are the distances of the object and image from the centre of the lens and *f* is the focal length.

If we rewrite $a = z + f$, and $b = z' + f$, we get the following equation:

$$\frac{1}{z+f} + \frac{1}{z'+f} = \frac{1}{f} \tag{2}$$

which can be used to calculate the following interesting properties.

2.3.1. Field of view (FOV)

The magnification of a lens system can be defined as:

$$m = \frac{x'}{X} = \frac{b}{a} = \frac{z'+f}{z+f} \tag{3}$$

If we assume that $z >> f$, then it follows that $z' << f$. Taking this into account we can simplify to

$$m \approx \frac{f}{z} = \frac{z'}{f} \tag{4}$$

From this we can see that to change the magnification, one can either change the distance to the object, or change the focal length.

When you calculate the field of view, you are actually calculating "how large an object can I fit onto my image sensor with this magnification?" Let *D* be the width of the image sensor, then

$$m = \frac{x'}{X} = \frac{D}{FOV} \tag{5}$$

$$\leftrightarrow \; FOV = \frac{D}{m} \tag{6}$$

The field of view thus depends on the distance to the object, the image sensor size, and the focal length. In industrial machine vision applications, the distance to the object is usually kept constant. After determining the distance to the object, and after the sensor size of the camera is known, the focal length needed to give the desired FOV is calculated.

2.3.2. The f-number

A larger lens captures and focuses more light and thus produces a brighter image. The total amount of light that passes through the lens system and hits the sensor can be adjusted by placing a variable aperture somewhere in the lens system.

We saw earlier that the magnification depends on the focal length. A longer focal length projects light gathered onto a larger area. Thus the total light intensity projected onto the image sensor depends on both the size of the lens/aperture and the focal length. The f-number is defined as:

$$n_f = \frac{f}{d} \tag{7}$$

where *f* is the focal length and *d* is the diameter of the aperture. Lenses with the same f-number project the same amount of light onto the image sensor. A larger number gives darker images.

The total amount of light hitting the sensor depends also on the amount of time it is exposed to light i.e. the shutter speed of the camera. A lens with smaller f-number means that you get the same amount of light faster, and therefore photographers usually interpret the f-number as a measure of lens *speed*. The advantage of shorter exposure times is that image blur due to movements is reduced.

2.3.3. Depth of focus and depth of field

According to the thin lens equation, only objects precisely at distance $f + z = a$ from the lens will form a sharp image at the image sensor at distance $f + z' = b$ from the lens. If we move the image sensor by a distance $\Delta z'$ then the point at distance a from the lens will appear as a circle with diameter ε on the sensor (Figure 10.)

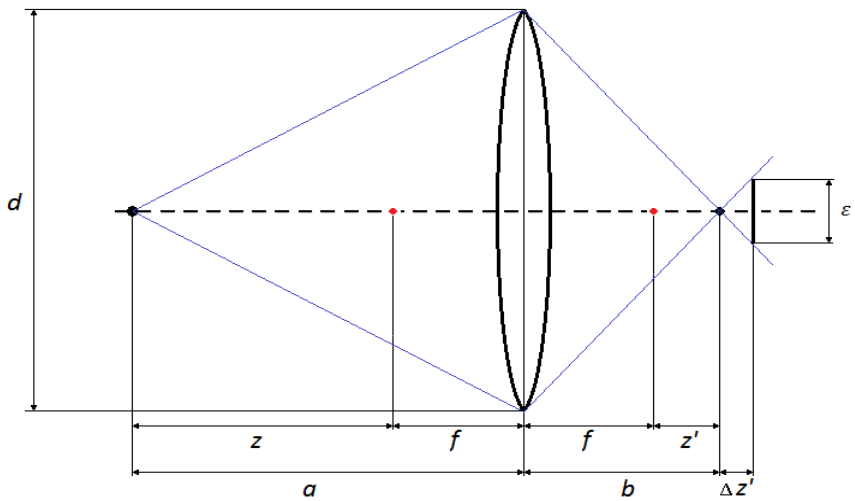


Figure 10. A point at distance $f + z = a$ from the lens forms a circle at the image plane located at $f + z' + \Delta z'$. Note that if we reduce the lens diameter d, then also the circle diameter ε will get smaller.

From the figure we get the following:

$$\frac{\varepsilon}{\Delta z'} = \frac{d}{f+z} \tag{8}$$

$$\leftrightarrow \ \Delta z' = \frac{\varepsilon(f+z)}{d} \tag{9}$$

If we substitute $d = f / n_f$, we get

$$\Delta z' = \frac{n_f * \varepsilon(f+z)}{f} = n_f * \varepsilon\left(1 + \frac{z}{f}\right) \approx n_f * \varepsilon(1+m). \qquad (10)$$

$\Delta z'$ is called the *depth of focus*, and this formula tells us how much we should move the image plane to get an ε-sized circle when projecting a point. While ε is kept smaller than the size of one pixel, the resulting picture is not blurred. We can see that the depth of focus is directly proportional to the f-number.

If we had changed the problem a bit and calculated Δz instead of $\Delta z'$, we would end up with a formula telling us how much the distance to the object could change without blurring the image. This property is called the *depth of field*. Both depth of field and depth of focus tell us the same thing, namely what are the tolerances for the distance to the object to still get a sharp image. An example of the dependence between the depth of field and the f-number is shown in Figure 11.

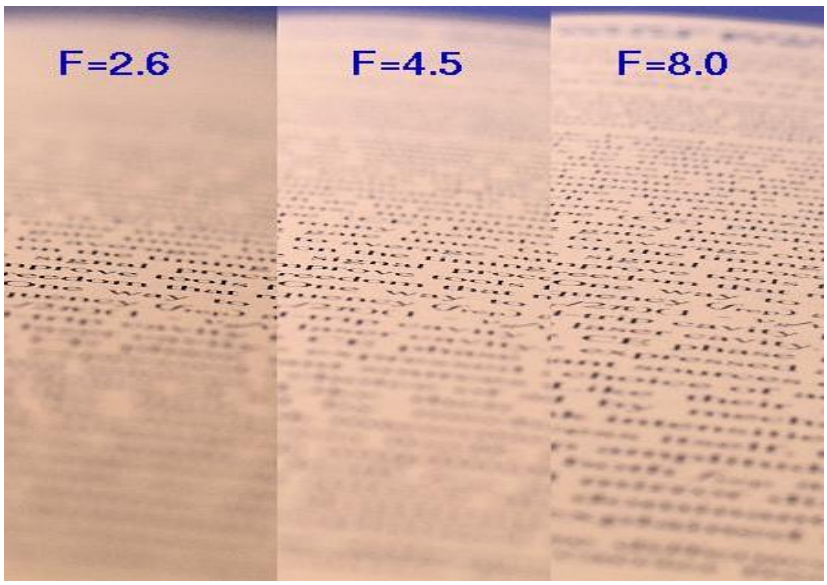


Figure 11. Smaller f-number means a faster lens, but also a shallower depth of field (Picture: focoz.com)

2.3.4. Lens selection

There are no high-end requirements on the lenses in this application. The lenses should be suitable for the image sensor size and resolution, and these are within values normally used. The distance to the material is large enough not to cause troubles, and exceptional image quality or high speed is not required.

A focal length of 8mm would give the optimal FOV, but since the cameras have unnecessarily high resolution, 6mm lenses were chosen. This projects the inspection area to a smaller area on the sensor, and the number of pixels that are of interest is thus reduced.

The lenses used are Ricoh FL-HC0612A-VG standard machine vision lenses. This model is suitable for sensor sizes smaller than 1/2-inch. The sensor size of our camera is 1/1.8-inch, but as mentioned above, a focal length of 6mm was selected so that we can project the ROI onto the centre of the sensor and ignore the edge pixels. The minimum f-number is 1.2, which is slightly lower than for the cheapest lenses. This gives a bit more flexibility when setting the exposure time of the camera.

2.4. Image processing methods

This section will describe the image processing methods that are used in this work. Details on how the different methods are used will be discussed in the chapters that present the complete algorithms. Comments on performance will also be given in those chapters.

The FZ5 algorithm uses a dilation filter to pre-process the images. This filter is one of many that use mathematical morphology to modify images. The first section will briefly explain the two most common morphological filters, namely the dilation and the erosion filters. After the pre-processing is performed, the images are measured with two different methods that Omron call “Gravity and Area” and “Defect”. These methods are described in their own sections.

The algorithm developed in Matlab transforms the image into its frequency representation using the Fourier transform. Using a series of calculated feature values, the image is then judged with the help of fuzzy membership functions. These topics are discussed in the two last sections of this chapter.

2.4.1. Morphological filters

Mathematical morphology can be used to alter the shape of spatial structures. Morphological operations modify shapes in an image *g(r,c)* by the use of a second image *s(i,j)*, which is defined in a finite region *S*. The second image *s(i,j)* is called a structuring element. The two most common morphological operations are dilation and erosion.

The dilation operation is defined as a *Minkowski addition* with a transposed structuring element:

$$\mathrm{g} \oplus \check{s} = \max_{(i,j)\in S}\{\mathrm{g}_{r+i,c+j} + s_{i,j}\} \tag{11}$$

The typical choice for the structuring element is a flat image with a constant value of zero. Then the definition reduces to

$$\mathrm{g} \oplus \check{s} = \max_{(i,j)\in S}\{\mathrm{g}_{r+i,c+j}\} \tag{12}$$

This means that for every pixel in the image *g(r,c)* we replace the current pixel value with the maximum value in the neighbourhood defined by the region *S*. This simplification is therefore often called a *maximum filter*. A flat structuring element is so commonly used, that dilation filters and maximum filters are usually considered the same. The "dilation" filter in the Omron library is in fact a maximum filter. (Steger, Ulrich & Wiedemann, 2008)

The erosion operation is defined as a *Minkowski subtraction* with a transposed structuring element:

$$\mathrm{g} \ominus \check{s} = \min_{(i,j)\in S}\{\mathrm{g}_{r+i,c+j} - s_{i,j}\} \tag{13}$$

Again, using a flat structuring element with a constant value of zero gives us the *minimum filter* which is a special case of the erosion filter, but usually considered the same. (Steger, Ulrich & Wiedemann, 2008)

The dilation filter is most commonly used to enlarge bright regions in the target image *g(r,c)*. In the same way, erosion filters are usually used to enlarge dark regions.

2.4.2. Omron's Gravity and Area method

The Gravity and Area method found in Omron's library calculates the area and the centre of gravity of a region in the image (Omron, *Vision Sensor-FH/FZ5 Series Vision System Processing Item Function Reference Manual)*. First, a *thresholding* operation is applied to the image to create a binary image. A thresholding operation defines an interval of intensity values that is included in the output region. Pixels in the output region gets the value 1, and are displayed as white, while pixels outside the output region gets the value 0 and are displayed as black. The simplest thresholding operations, as the one used in the Gravity and Area method, let the user set the limits of the interval to constant numbers that are used on every pixel in the image. More advanced methods tries to calculate optimal threshold values from the image data, and some calculates new values for every pixel based on local intensity values.

After the thresholding, the method calculates the area and the centre of gravity of all white pixels. These two values are special cases of a group of region features called *image moments*

$$m_{p,q} = \sum_{(r,c)\in R} g_{r,c} r^p c^q \tag{14}$$

where p and q are integers, r and c are pixel coordinates and $g_{r,c}$ is the grey value of the pixel at position (r,c). By calculating m_{00} one gets the image area. In the case of a binary image where grey values are either 0 or 1, this corresponds to the total number of pixels with the value 1. The ratio $m_{1,0}/m_{0,0}$ gives the x-coordinate of the centre of gravity, and correspondingly $m_{0,1}/m_{0,0}$ gives the y-coordinate.

After calculating the area and the centre of gravity, the method checks if they are within the limits set by the user.

2.4.3. Omron's Defect method

The Defect method is meant for detecting spots on uniformly coloured materials, and it does this by searching for local deviations in intensity. It divides the image into regions, whose size can be selected, and measures the mean intensity value in all regions. A defect value is calculated from the difference between neighbouring regions. The smaller the regions, the smaller defects can be detected by the method, but it also runs slower since the number of regions is then increased. (Omron, *Vision Sensor-FH/FZ5 Series Vision System Processing Item Function Reference Manual)*

2.4.4. The Fourier transform

Any continuous signal in the time domain can be exactly reconstructed using an infinite number of sine and cosine waves with different frequencies and magnitudes. The Fourier-transform is a mapping of a signal into its frequency representation. The Fourier transform for a one dimensional continuous function *f(x)* is defined as

$$F(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi ikx}\, dx \tag{15}$$

The transformation into the frequency domain preserves all information about the signal. The result is simply another representation of exactly the same signal, and it can be transformed back to the original domain by an inverse transform:

$$f(x) = \int_{-\infty}^{\infty} F(k) e^{2\pi ikx}\, dk \tag{16}$$

When dealing with digital images, which are finite, discrete and real signals in the spatial domain, we do no longer need an infinite series of frequencies to fully represent them in the frequency domain.

When a continuous world image is sampled by a digital camera, frequencies outside the interval [$-f_s/2$, $f_s/2$] where f_s is the sampling frequency, are aliased onto frequencies with-

in this interval. This phenomenon causes image distortions, and therefore frequencies outside this interval are usually removed using anti-aliasing filters, before the image is captured. When dealing with signals in the spatial domain, one usually defines a signal with frequency 1, to be a signal that has one period within the whole series of samples. Thus, the sampling frequency becomes equal to the total number of samples *N*, and we can rewrite the above interval as [-*N/2, N/2*].

The *Discrete Fourier Transform* (DFT) can be used to transform discrete signals to their frequency representation:

$$X_k = \sum_{n=0}^{N-1} x_n \, e^{-2\pi ikn/N} \tag{17}$$

Here, X_k is the value of the *k*:th frequency in the frequency domain, x_k is the value of the *n*:th sample in the time domain, and *N* is the total number of samples. X_k is a complex number that represents both the magnitude and phase of the frequency component. The magnitude of a complex number is computed by $M = \sqrt{R^2 + I^2}$, where *R* and *I* are the real and imaginary parts respectively. The phase angle is computed by $\varphi = \tan^{-1}(\frac{I}{R})$. If a finite series with *N* values is transformed, the formula of the transform becomes periodic with period *N*, i.e. $X_k = X_{k+N}$. Therefore, to avoid negative indexes, one can compute the frequencies in the interval [0, *N*] instead of [-*N/2, N/2*]. The negative frequencies are now contained in the upper half of this interval.

Actually, when the original signal is real valued, the transform contains redundant information, since $F(-k) = \overline{F(k)}$. Often in image processing, only the magnitudes of the frequency components are used in analysis, and from the above fact one can see that the magnitude of the negative frequencies are exactly the same as for their positive counterparts. Some implementations of the transform take this into account, and only calculate half of the transform. These implementations are called real valued Fourier transforms. (Steger, Ulrich & Wiedemann, 2008)

Images are 2-dimensional signals, and are therefore transformed using the following 2-dimensional DFT

$$F(k,l) = \sum_{y=0}^{N-1}\sum_{x=0}^{M-1} f(x,y) * e^{-i[\left(\frac{2\pi k}{N}\right)x+\left(\frac{2\pi l}{M}\right)y]} \quad (18)$$

where *N* is the number of pixel rows and *M* is the number of pixel columns. The result is an *NM* matrix of complex numbers.

Computing the Fourier-transform directly by the above formula is a time consuming process. The time is directly proportional to the second order of the image matrix size, i.e. $O((NM)^2)$ for *N-by-M* matrixes. Several methods that compute the same formula in a more efficient way have been developed and they are known as Fast Fourier-Transform (FFT) algorithms. These algorithms require $O(N\log_2 N)$ time to compute in the one dimensional case (Oppenheim, Schafer & Buck; 2000). The 2-D transform is equivalent to first taking the one dimensional transform for every row, *N* and then for every column *M*. This adds up to $O(M(N\log_2 N) + N(M\log_2 M))$, which is $\approx O(2N^2\log_2 N)$ if *N* and *M* are approximately the same size. This can save a lot of time for large image sizes.

An example of a frequency magnitude spectrum is shown in Figure 13. The image in Figure 12 is transformed into the frequency domain using a 2-D FFT. The result has the negative frequencies after the positive frequencies, and hence we shift them to their correct places so that the origin is in the middle of the resulting matrix. Finally, the magnitudes are computed and shown in logarithmic scale in Figure 13.

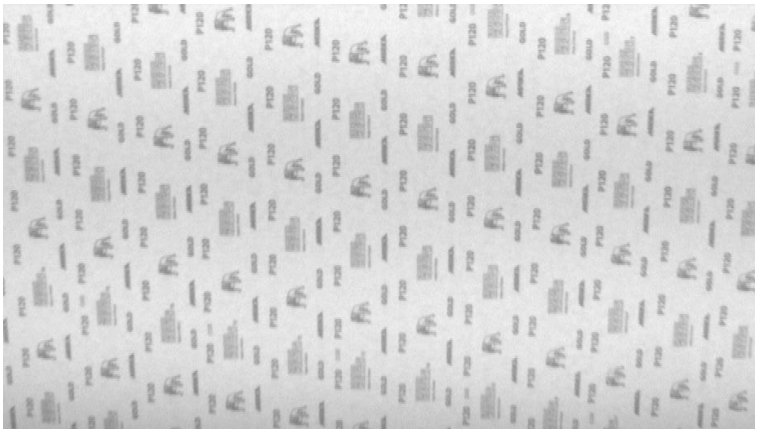


Figure 12. The original test image in the spatial domain.

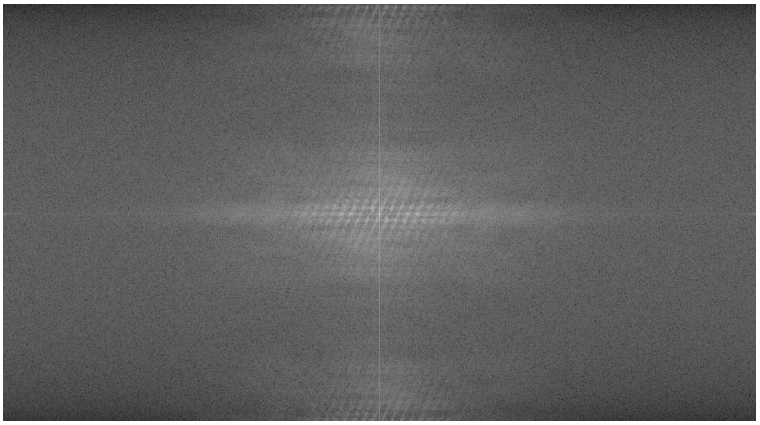


Figure 13. The magnitude spectrum of the test image in Figure 12. Note the pattern of lines that are perpendicular to the main direction of the texture in the original image. A logarithmic scale is used for displaying the result, since the low frequencies, especially the DC component, have much higher values than the high frequencies.

2.4.5. Feature analysis with fuzzy membership functions

In this work we are searching for unexpected faults in patterns with the help of outlier detection. First, one tries to find a set of features that describe the faultless pattern, but that are sensitive to the changes one tries to detect. Feature data is then collected from a series of faultless sample images and used to train the detector. After the training phase is done, the detector compares input data to learned values to see if they are similar. The output from this kind of detector is only a judgment of the similarity to trained values, no classification of the detected defect is done.

One way for the detector to perform the comparison is to calculate crisp limits for the feature values from the training data. If a feature value is inside the limits, the value is considered normal. Mathematically explained, the set of data points F for a given feature has a *membership function* associated with it, that maps a value to its degree of membership in the subset A of "normal values". In the case of crisp limits a and b (one dimensional case) the membership function is

$$\mu A(x) = \begin{cases} 1, & a < x < b \\ 0, & otherwise \end{cases} \tag{19}$$

i.e. the function maps the data point x in F to the set {0,1} depending on whether the point is within the limits or not.

While this approach might work well in some cases, it is far from ideal. Consider a case where one out of ten feature values is barely outside the limits while the others are perfectly normal, versus the case that all ten feature values are barely inside the limits. Is it correct to judge the first case as faulty, while the other is judged as completely fine?

Instead of mapping to either zero or one, a fuzzy membership function maps a point to the interval [0, 1]. Thus, a point can have a partial membership in A. A fuzzy membership function can have any form, but commonly used functions are triangular, trapezoidal, Gaussian, sigmoidal, and polynomial based membership functions.

In this work we use Gaussian membership functions. They are slower to compute than the piecewise linear (triangular and trapezoidal) functions, but they describe the input data better. One example of this is the fact that the Gaussian membership function never reaches zero membership. Since all inputs to the detector come from images of the same material that was used for training, it is logical that no input gets zero similarity.

The Gaussian membership function has a midpoint α, and the membership values decrease as we move away from the midpoint. The rate of decrease depends on a second parameter β. The formula for the Gaussian membership function is

$$\mu A(x) = e^{-\beta(x-\alpha)^2} \tag{20}$$

We can modify the membership function so that it has different β-values for the right and left slopes. We can also specify an interval where the memberships are constant one, instead of a single midpoint. If we define α_l and α_r for the left and right limits of this interval, and β_l and β_r as the slope values for the left and right slopes respectively, we get the membership function

$$\mu A(x) = \begin{cases} e^{-\beta l(x-\alpha l)^2}, & x < \alpha_l \\ 1, & \alpha_l < x < \alpha_r \\ e^{-\beta r(x-\alpha r)^2}, & x > \alpha_r \end{cases} \tag{21}$$

This is the type of membership function used in this work. An example is shown in Figure 14.

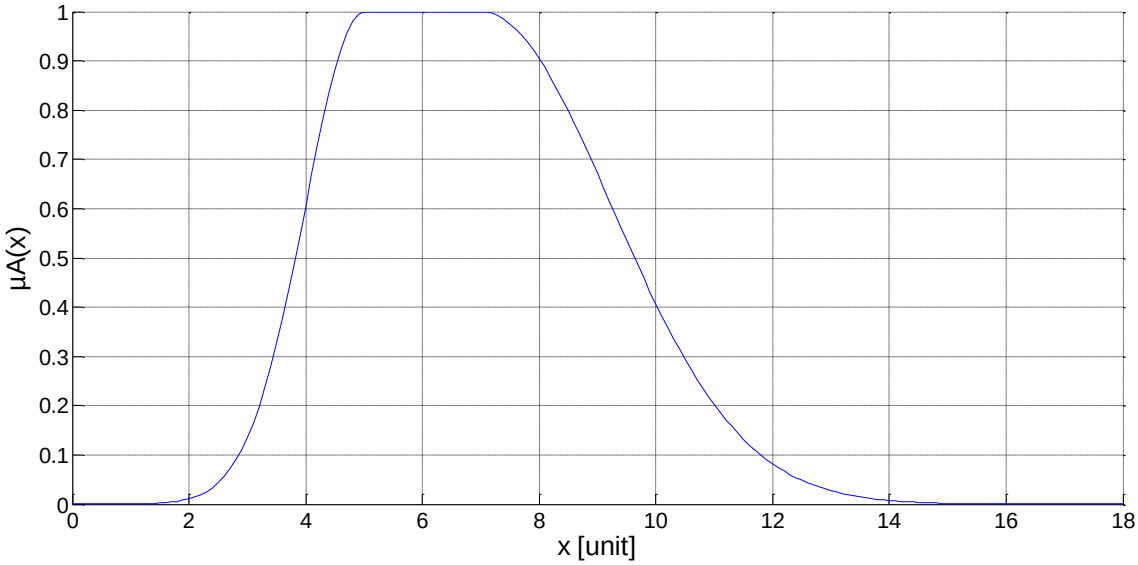


Figure 14. Generalized Gaussian fuzzy membership function with $\beta_l = 0{,}5$; $\alpha_l = 5$; $\beta_r = 0{,}1$ and $\alpha_r = 7$.

The detector computes the four parameters for the membership function from the training data. The alpha values can be computed by requiring that a certain percentage of data points should be located within the interval. We define a midpoint, usually the mean value, and then include the correct percentage of points closest to the midpoint of the interval.

The data points that are to the left and right of the alpha-interval are used to compute the left and right beta value, respectively. The beta values should be dependent on the standard deviation of the data points, so that more spread out feature values give longer slopes than tightly gathered feature values. It is common to write the Gaussian function in the form

$$f(x) = e^{\frac{-(x-\alpha)^2}{2\sigma^2}} \tag{22}$$

where σ is the standard deviation of the function. From this we see that the beta value in the membership function should be set to $\beta = 1/(2k\sigma^2)$, where *k* is a constant. A larger value of *k* gives longer slopes.

After the detector has been trained, it has membership function for every input feature. An input vector is mapped trough the functions and we get a membership value for each feature. These should be combined into a single membership value that is assigned to the input vector, and depending on this value the detector decides whether the input is faultless or not. In this work, the values are combined by taking their product.

3. IMPLEMENTATION OF ONLINE MEASUREMENTS WITH THE FZ5-SYSTEM

3.1. Goals and requirements

With the Omron FZ5-system we are trying to build a complete solution for performing real-time measurements. The system must be able to handle all different materials used in the machine, it has to work in an industrial environment, communicate with the machines logic controller, and it should have a user interface that is easy to use.

Four different materials are frequently used. Abranet and Autonet are net-type materials, and Gold and QSilver are paper-type materials. The older version of QSilver, Mirka Silver, is also sometimes used. Images captured of these different materials are shown in Figures 15 - 17.

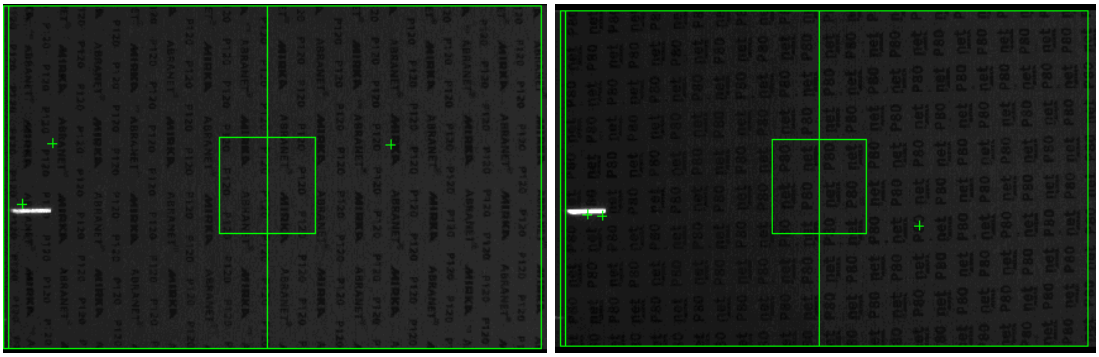


Figure 15. Images captured of the two net-materials, Abranet to the left and Autonet on the right.

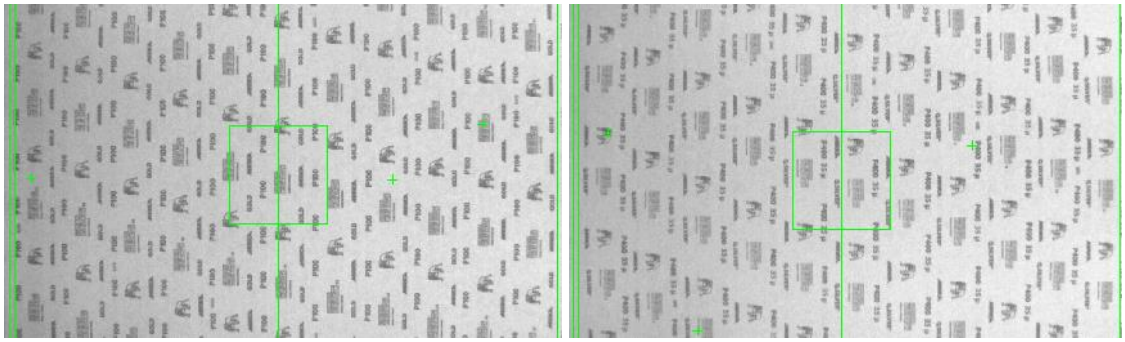


Figure 16. Images of the two most common paper-materials, Gold to the left and QSilver on the right.

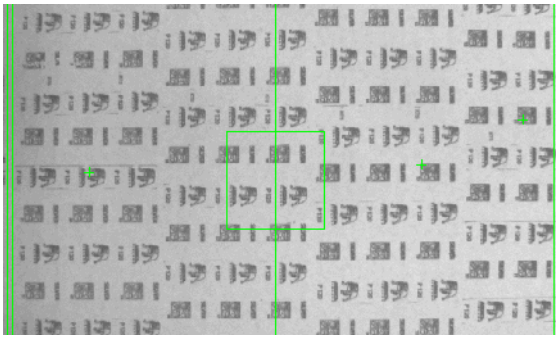


Figure 17. The older Silver paper has prints that differ from the newer QSilver.

The environment is very dusty, so the cameras are mounted inside metal boxes to protect from some of the dust, and further, pneumatic hoses are connected to nozzles aimed at the camera lenses. A continuous, low flow of air keeps most of the dust away. Another environmental problem in this case is that the cameras can see through the net-materials. Since they are pointed upwards, lamps in the factory ceiling are visible as bright spots in the image (see Figure 15). These spots must be removed in software since the machine operators did not want us to place an enclosure behind the material.

The system performs inspections only when requested from the PLC. A digital output on the FZ5-system is activated when it is ready to accept inspection commands. The controller sends a command to the FZ5-system's parallel port to trigger an inspection. If a defect is detected, a digital signal is activated that causes the PLC to stop the machine.

The user interface is shown in Figure 18. It contains runtime information that can be used to check the performance, but in normal operation no actions are required. Inspections are activated manually with a button in the machine's HMI, and the same button can be used to acknowledge detected defects. Omron uses the abbreviations "OK" and "NG" for the inspection results. The camera and lighting setup is shown in Figure 19.

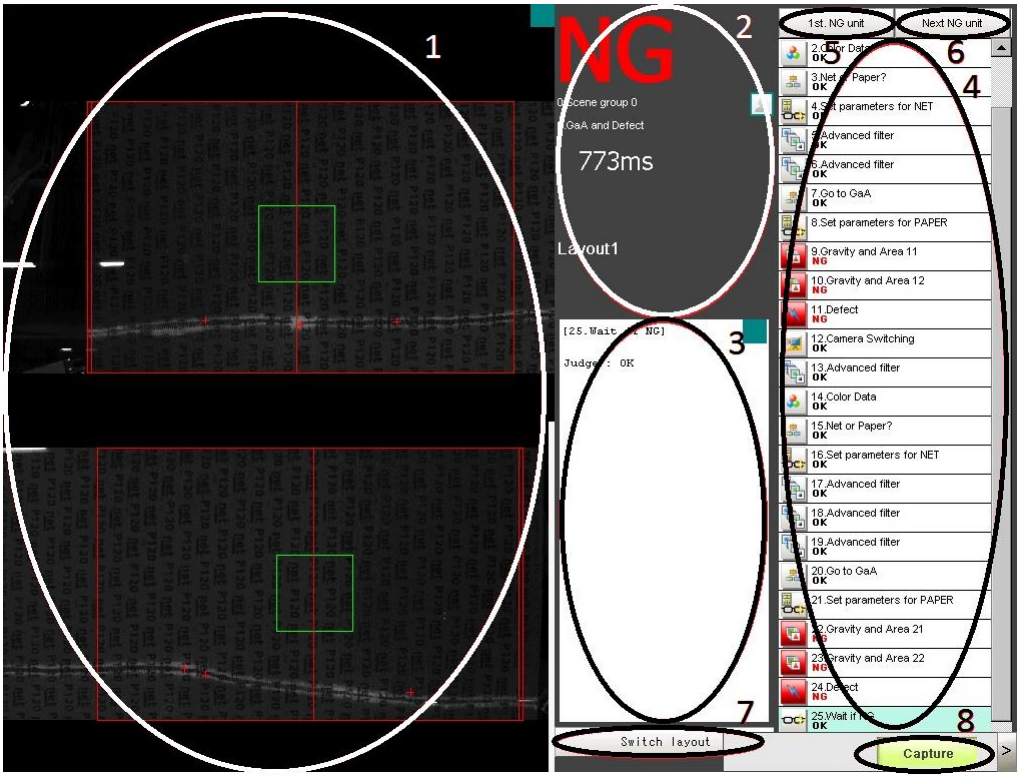


Figure 18. The user interface. 1 – Camera images with ROIs for the different methods. 2 – Status window with information about loaded programs, inspection time and result. 3 – Method data window. Displays information about the method selected in the method flow list. 4 – Method flow list shows the methods in the processing sequence. The icon for a method that returns "NG" turns red. 5 and 6 – Buttons for finding methods that returns the result "NG". 7 – Button for switching the layout from this one used while running, to the settings and debug layout. 8 – Button for manual image save.

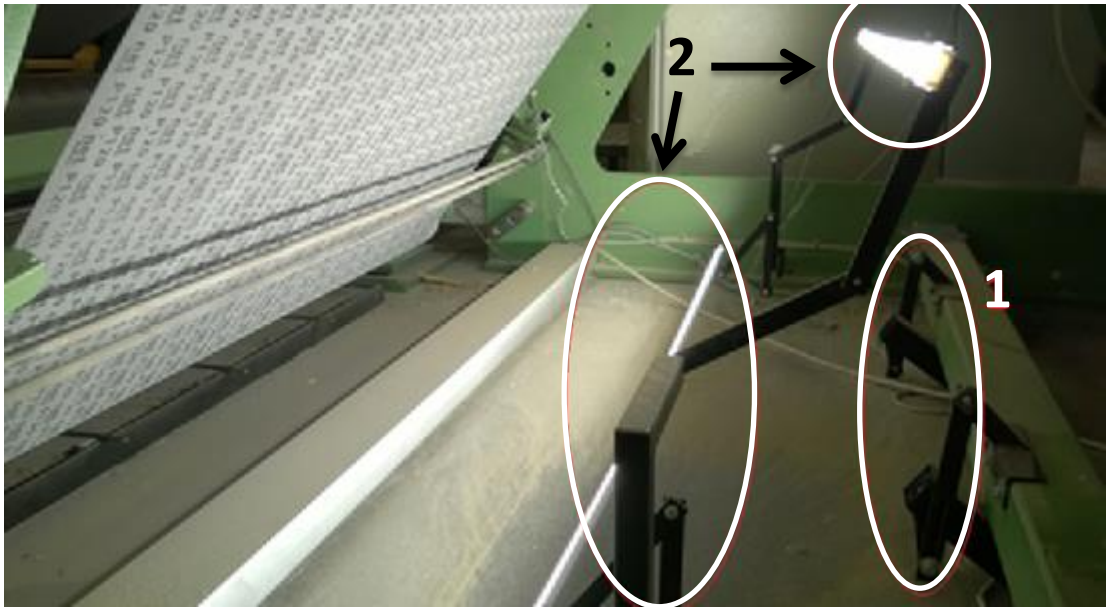


Figure 19. The setup with the cameras (1) near the floor and line lights (2) both above and under the cameras. The nozzles for cleaning the lenses are not visible in this image.

3.2. Presentation of the FZ5 algorithm

This section presents the algorithm that is used in the FZ5-system to search an image for defects. The section includes a general discussion of the algorithm and comments on performance and tuning of parameters. Explanations on separate function blocks, along with an image generated by the FZ5-systems flow viewer can be found in Appendix A.

The flowchart for the FZ5 algorithm is shown in Figure 20. First, the image is pre-processed with a 9 times 9 dilation filter. After this, the mean intensity of the image is calculated. This value is used to decide which material we are dealing with and to compensate for intensity variations. The algorithm only differs between two classes of material, net- or paper type materials.

After deciding which class of material is used, the algorithm calculates and sets the parameters for the “Gravity and Area” and “Defect” methods which are used to analyse the image for defects. The parameters calculated are the binary limits for the “Gravity

and Area" method. They are set to a percentage of the mean intensity value. Limits for the calculated area, and the output from the "Defect" method are set only depending on the type of material. If the material is of net-type, the bright spots are removed by over-writing them with a lower intensity value. The image is then analysed with "Gravity and Area" and "Defect" methods.

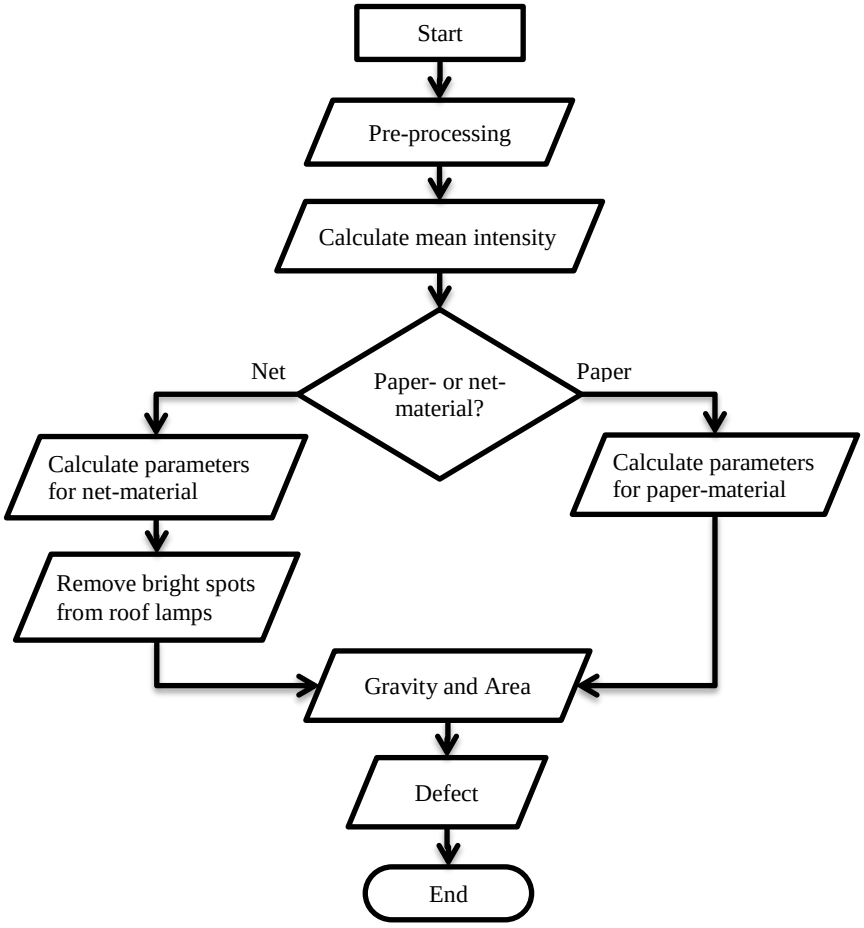


Figure 20. Flowchart for the FZ5 algorithm.

The ideal case for analyzing an image for defects with the "Gravity and Area"- method, would be to find binary limits that separates the background from the rest of the image. Then the area should be equal to the area of the prints, otherwise we know that there are also other unwanted details in the image.

Several problems arise. First, the illumination is not perfectly uniform. This causes the background to include quite a broad range of intensity values. Prints in the brighter parts

of the image are brighter than the darkest parts of the background. The problem could have been fixed with locally adaptive thresholding, which is however not available in the Omron library. Secondly, the method is of course highly sensitive to changes in illumination. The algorithm tries to adjust the binary limits depending on the mean intensity value in the image, so that the background always lies within the limits, and this works quite well. However, the big problem is that the edges of the prints are not perfectly sharp. A small change in the threshold value changes the area of every single print. The impact on the total area of the image might be a few thousands of pixels. Allowing the area value to change this much would make the method totally useless. A solution to this problem could have been to use edge detectors to segment the prints instead of thresholding. Edge detectors based on image gradients have better abilities to precisely locate the edge even if it is blurred. Another possibility would be to use labelling and blob analysis to inspect every print independently. The tools needed for this are available and could give good results, but would demand a system with more computing power.

In this case, a dilation filter is used instead to suppress the printings, and we choose binary limits so that the whole image should be black. Defects that are large enough not to be removed by the dilation are detected. In order to slightly compensate for the non-uniform illumination, two “Gravity and Area” blocks are used, one for the brighter part of the image and one for the darker.

The use of the dilation filter also enables us to use the “Defect” method, which is designed for detecting spots on uniformly coloured backgrounds. Yet another advantage is that the differences in prints for different materials are not that visible anymore and we can limit ourselves to only two classes of material. This simplifies the classification step to a simple intensity mean value calculation.

The mean value calculation is performed by a block named “Color Data”. It can be used to analyse colours, but when instead the input is a monochrome image, it calculates the mean value and standard deviation of the intensity values. The standard deviation could have been used to detect defects, and it was actually included at first. However, during

testing it was found that all defects that were detected by this parameter were also detected by the other two methods, so it was removed.

The program starts by reading the images from both cameras. The exposure time and other camera settings are set in such a way that it is suitable for both the darker net-materials and the brighter paper materials. High speed version FH-systems can perform parallel processing of the two images, but this cannot be done with the FZ5-L355. The algorithm is therefore first run on the first camera image, and then on the other image. A “Camera switching” block is used to switch to the other camera image when the first one have been processed.

Measurements are stopped by the PLC if a defect is detected. The digital signal that indicates that a defect has been detected is toggled at the end of the program. To ensure that the image that caused the stop remains in the monitor, a wait statement is placed in a macro and executed by the FZ5-system when a defect is detected. Without this statement, the system would start the next measurement and read in new images from the cameras before the PLC has time to react. This would overwrite the images that caused the stop.

4. OFFLINE DEVELOPMENT OF FREQUENCY SPACE APPROACH

The algorithm implemented on the FZ5-system relies on intensity changes in the material. It has no chance of detecting defects that have the same intensity as the surrounding material. Such a defect is shown in Figure 21. As a complement to the previous work, research on texture analysis was done to see if a reliable method could be found that detects changes in the texture, i.e. the prints on the backside of the material.

This part of the work is only about testing another approach. Some simplifications have been made compared to the online system. For example, the images have been cropped so that we get rid of the darker regions near the borders, and also the bright spots from the ceiling lamps. Further, only two different materials are tested, Autonet and Gold.

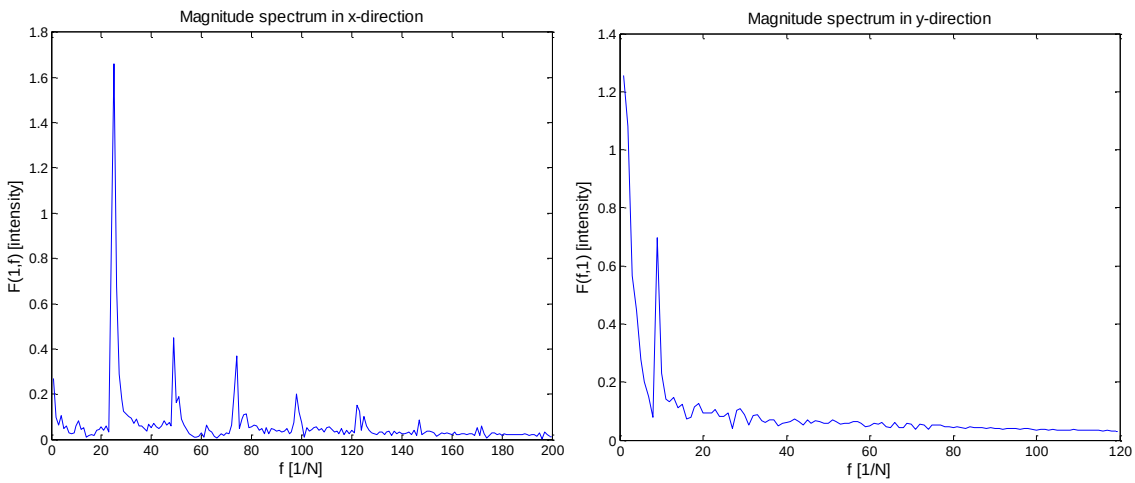
The periodic prints on the material, implies that features in the frequency spectrum could be useful. As mentioned in the related work section, CH Chan and Pang GKH at the University of Hong Kong used the Fourier-transform to detect defects in fabric. Their algorithm serves as a basis in my efforts to detect changes in the prints on MIRKA's abrasives.

Figure 21. Folds in the material cannot be detected by the FZ5 algorithm. The contrast in this image has been enhanced by processing.

4.1. Feature selection

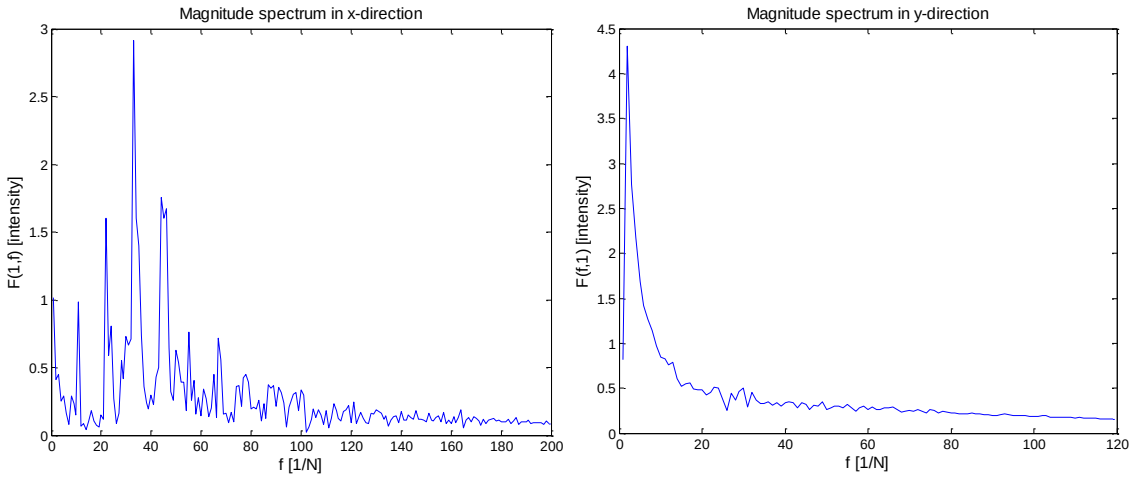
What makes features in the frequency space particularly interesting in this case is the translation property of the Fourier transform. It states that a shift of the function in time (or spatial) domain only affects the phase of the frequency content. The magnitude spectrum remains the same. This is only partially true when analysing a material that moves past a camera, since the image is only defined within a finite region and the borders of the image causes ringing at higher frequencies. Still, the magnitude spectrum is almost translation invariant and contains valuable information about the texture.

In Matlab, the 2-D FFT can be easily computed by the fft2() function. One gets the magnitude spectrum by using the abs() function on the result. These values are further divided by the image size, to bring down the DC-component, $F(1,1)$, within the interval [0, 255] (Matlab indexes starts from 1). The magnitude spectrum in both the x- and y-direction for a sample image of both Autonet and Gold is plotted in Figure 22 and 23. Due to its large magnitude, $F(1,1)$ is omitted from the plots. Including this value decreases the visibility of the small differences in the rest of the plot, even if using a logarithmically scaled y-axis.



(a) (b)

Figure 22. Frequency magnitude spectrums in (a) the x-direction and (b) the y-direction for Autonet.



(a) (b)

Figure 23. Frequency magnitude spectrums in (a) the x-direction and (b) the y-direction for Gold.

Chan and Pang used 7 features in their design. The first feature is the DC-component, which corresponds to the mean intensity in the image. The other features are the frequency of the first peak in the spectrum, the magnitude of this peak divided by the mean intensity, and the sum of all magnitudes up to this frequency divided by the mean intensity. These three features are calculated in both x- and y-directions.

The frequency of the prints on the backside of abrasives is lower than the frequency of yarns in fabric. As can be seen from Figure 22, the spectrum of Autonet shows a strong peak at a certain frequency and then several decaying peaks at the harmonics of this frequency. The features used by Chan and Pang could be suitable to this material by using the peaks at f = 25[1/N] in the x-direction and f = 9[1/N] in the y-direction. The Gold material however, which has more detailed printings, produces a messier spectrum. The highest peak in the x-direction is at f = 33[1/N], but with several significant peaks also on lower frequencies. In the y-direction, the only significant peak in this spectrum is at f = 3[1/N], and when analysing several images, of which all are without defects, this peak disappears in some test image spectrums.

A set of features that was better suited for analysing also the Gold material was selected.

These features are used in the final version of my algorithm:

- In the x-direction:
  - The frequency of the *highest* peak, f_p, on the interval [2, 50].
  - The magnitude of this peak.
  - The sum of all magnitudes on the interval [2, f_p].
  - The standard deviation of the magnitude values on the interval [2, f_p].
  - The x-coordinate for the centre of gravity of the frequency distribution on the interval [2, f_p].
- In the y-direction:
  - The sum of all magnitudes on the interval [2, 27].
  - The standard deviation of the magnitude values on the interval [2, 27].
  - The x-coordinate for the centre of gravity of the frequency distribution on the interval [2, 27].

Hence, in total 8 features are used. The sum of the magnitude values is a measure of how much frequency power the image contains, the centre of gravity detects if the power distribution shifts within the interval, and the standard deviation is a measure of how smooth the spectrum is. Since sensitivity to changes in illumination should be avoided, the DC component is not included in the feature set, but used for normalisation. The spectrum is divided by the DC-component before computing the above features.

4.2. Presentation of the algorithm

The algorithm developed consists of two main parts, or functions. The first abra_fuzzyParams(), extracts features from faultless images and calculates the α and β values for the membership functions. This function takes three values as input. The first is an integer Nrules that tells how many different materials we are generating membership functions for. In this work this value is always set to 2. The second input is another integer, namely the k-value used to calculate the β values (see section 2.4.5). A value of

10 is used for k in this work. The third parameter is a Boolean plotON, which can be set to true if one wants the function to plot the generated membership functions. The output from this function is a struct variable that contains the midpoint for each feature, the offsets for the alpha values from this midpoint, and also the value of Nrules. This struct is used as input to the other function abra_measureImages(), which uses the information to analyse input images. It takes one image at a time, computes the features and then projects them through the membership functions. This function also takes a Boolean input plotON, which can be set to true if one wants to plot the result for each image. In addition, this function outputs a struct variable, which contains the memberships, processing time, and judge results for every image.

The flowcharts for the above functions are shown in Figures 24 and 25. In the following sections the different steps in the flowcharts are explained in detail. The Matlab codes for the functions are listed in Appendix B and C.

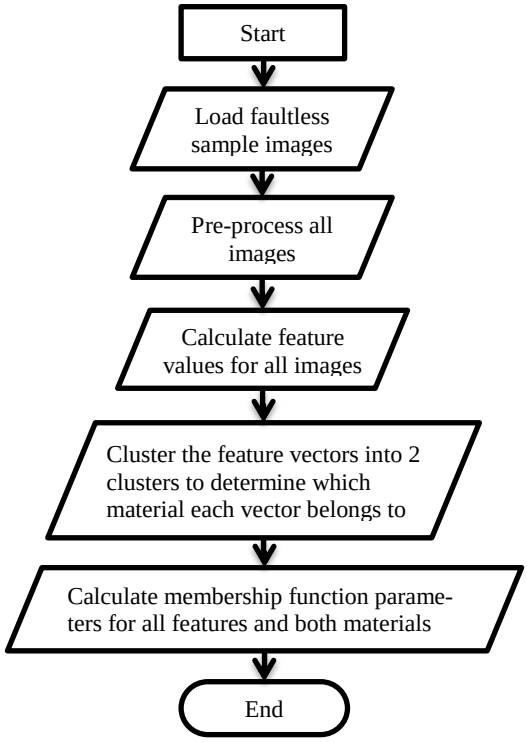


Figure 24. Flowchart for the function that computes parameters for Gaussian membership functions.

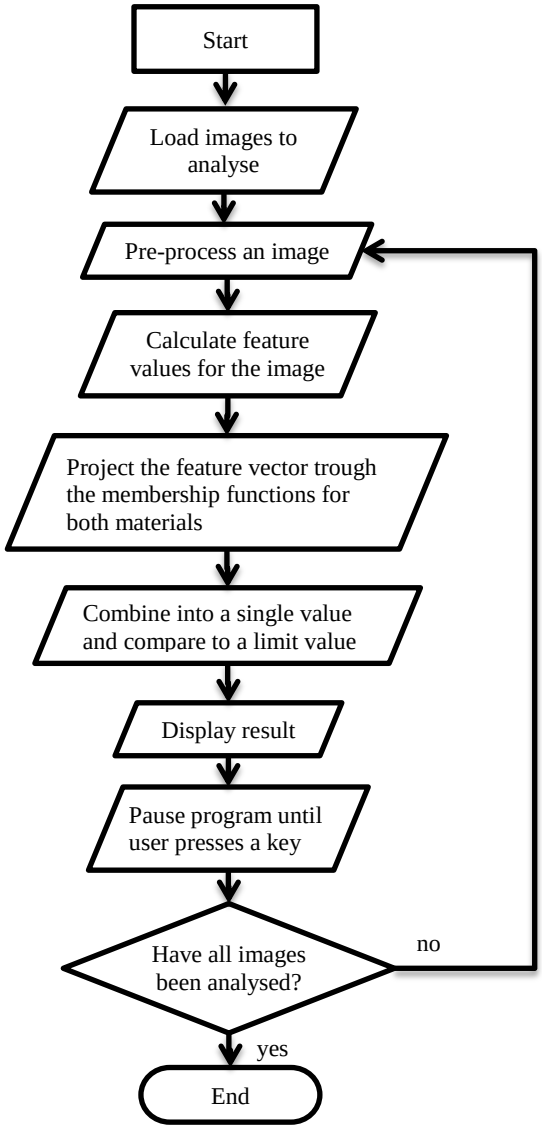
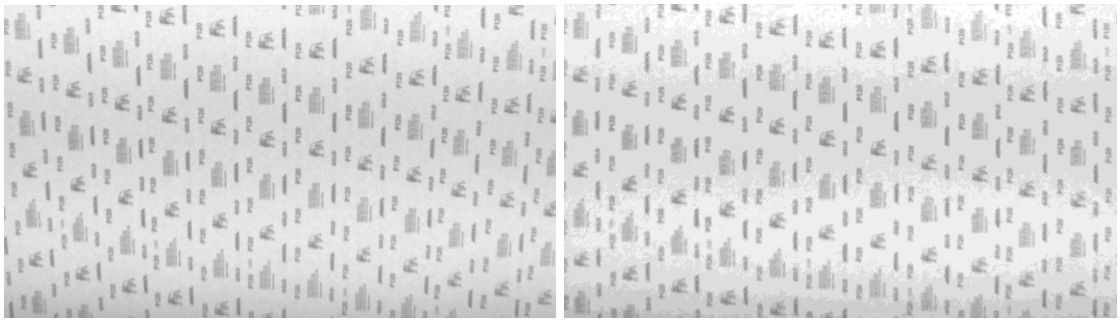


Figure 25. Flowchart for the function used to analyse images for defects.

4.2.1. Parameter calculation

A string variable named path is assigned with the path of the folder from where to read the sample images. A function getAllFiles() gets all filenames in the folder and stores them in a string vector imageList. Images of both materials are stored in the same list. Later, a clustering algorithm is used to decide which data vectors belong to which material. The variable imageList is passed as an input to a function named abra_features(). This function loops trough every image name in the list, and for every name it reads the image using Matlabs imread()-function, pre-processes it and calculates the feature values.

The pre-processing step crops the input image, performs trapezoidal correction, and filters the image with a Gaussian filter. Cropping is done to exclude pixels outside the ROI. The images used where not taken perfectly perpendicular to the material, and therefore gets a trapezoidal shape. This is corrected for before the image is smoothed. An example of this operation is shown in Figure 26. The better solution in an online version would of course be to place the cameras perpendicular to the material and omit the correction in software.



(a) (b)

Figure 26. Trapezoidal correction is used to get an image that looks as if the camera would have been located perpendicular to the material. The image before the correction is shown in (a), and the result is shown in (b). Note that in (a), the prints slides slightly inwards in the image as they move downwards.

The abra_features()-function returns a feature matrix FM that has eight columns, one for every feature value, and one row for every input image. After an image has been loaded, and pre-processed, the feature vector for that particular image is computed by the function freq_features(), and is then inserted into the feature matrix.

The freq_features()-function computes the magnitude spectrum of the image by first taking the 2-D FFT, divide the result by the image size, and then taking the absolute value. Dividing by the image size, i.e the total number of pixels, the values in the magnitude spectrum are scaled down to the interval [0, 255].

Before computing the features, the spectrum is normalised by dividing by the DC-component of the image. This somewhat compensates for illumination changes. Then the eight different features described in section 4.1 are computed. Before returning the values as a feature vector, some of them are multiplied by constants. This is only for the sake of nicer values to the human eye.

After all images have been processed, the feature matrix FM is returned to the main function, abra_fuzzyParams(). If every input image would have been of the same material, then it would be straightforward to use the whole matrix to calculate the alpha and beta values for the membership functions for this material. Parameters for different materials could have been calculated in separate runs of this algorithm. However, author's implementation of the fuzzy C-means (FCM) clustering algorithm was already available from a previous project. The use of this algorithm enables us to include images of several materials in the same run; the clustering algorithm takes care of the sorting before we calculate the membership function parameters. This stage is not essential to the performance of the defect detection, hence follows what is only a very brief description of the FCM algorithm.

The algorithm takes as input the feature matrix FM, the number of clusters c i.e. the number of different materials, an integer value m that is used as a weighting exponent when calculating memberships, and a real value e that is used as a stopping condition.

As output, we get a matrix U that contains the membership values between 0 and 1 for every row vector in FM in every cluster. We also output a matrix v that contains the midpoints for the clusters.

The matrix U is first initialized with random membership values. From these values we calculate the midpoints v of the resulting (random) clusters. The midpoint of a cluster is calculated as the mean value of the feature, but with every value weighted by its membership in that cluster. When we have calculated a set of midpoints, we can update U with values that depend on the distance to the midpoints.

The algorithm now checks how much U has changed, and if the change is smaller than the stopping condition e, the algorithm terminates. Otherwise it repeats the calculation of v and U. By iterating the updates of v and U several times, the midpoints move closer and closer to where there are high memberships, and the memberships calculated also becomes even higher as the midpoints move closer. Therefore, the midpoints will move towards dense collections of data points.

The performance of the FCM algorithm depends on the input data, and does not always give good results, but it is successful in separating the materials in this work. The computed memberships are used as a label for every row vector in FM telling which material it belongs to. Also, the computed cluster midpoints are used as alpha values in the next step.

The FCM algorithm assigns a membership value to each data point, which could actually be used for judgement instead of the values that are calculated later in this work. This is not done due to the fact that we do not want membership functions that depend on other materials. We are not trying to classify materials, but to detect abnormal values in a single material.

In this work, the midpoints from the clustering algorithm are stored and output in a variable named alpha. The two alpha values α_l and α_r that are needed are defined using offsets from the midpoint. These offsets are stored in variables named midL and midH for the low and high “midpoints” respectively.

These offsets and beta values are computed by the function calc_betas(). The function loops trough all materials (clusters), and all feature values for that material. First the offsets from the alpha values are computed. In this implementation, the intervals are set so that 75% of all feature values get memberships equal to 1. The rest of the values are used for calculating beta values. This stage simply searches for the 25% of the values that are furthest away from the midpoint in both directions and saves their distance from the midpoint in a separate array devPoints. The offsets for the alpha values are set to the minimum distances in both directions contained in this array. Then the Beta values are calculated from the points in devPoints according to the formula presented in Section

2.4.5. The first feature, the index of the highest peak, is not projected trough a Gaussian membership function so no parameters are calculated for this feature.

4.2.2. Defect detection

The images are loaded in pretty much the same manner as in the function for calculating the parameters. The difference is that, instead of collecting feature vectors for all images into an array, only a single image is loaded and analysed completely before the next one. Exactly the same pre-processing is performed on the images as by the function for parameter calculation. However, here it is not done by a separate function.

The function freq_features(), is called directly to calculate the feature vector for a single image. It is exactly the same function used when calculating parameters. Then it was called from the function abra_features() that collected the returned vectors into a matrix. This time that function is omitted.

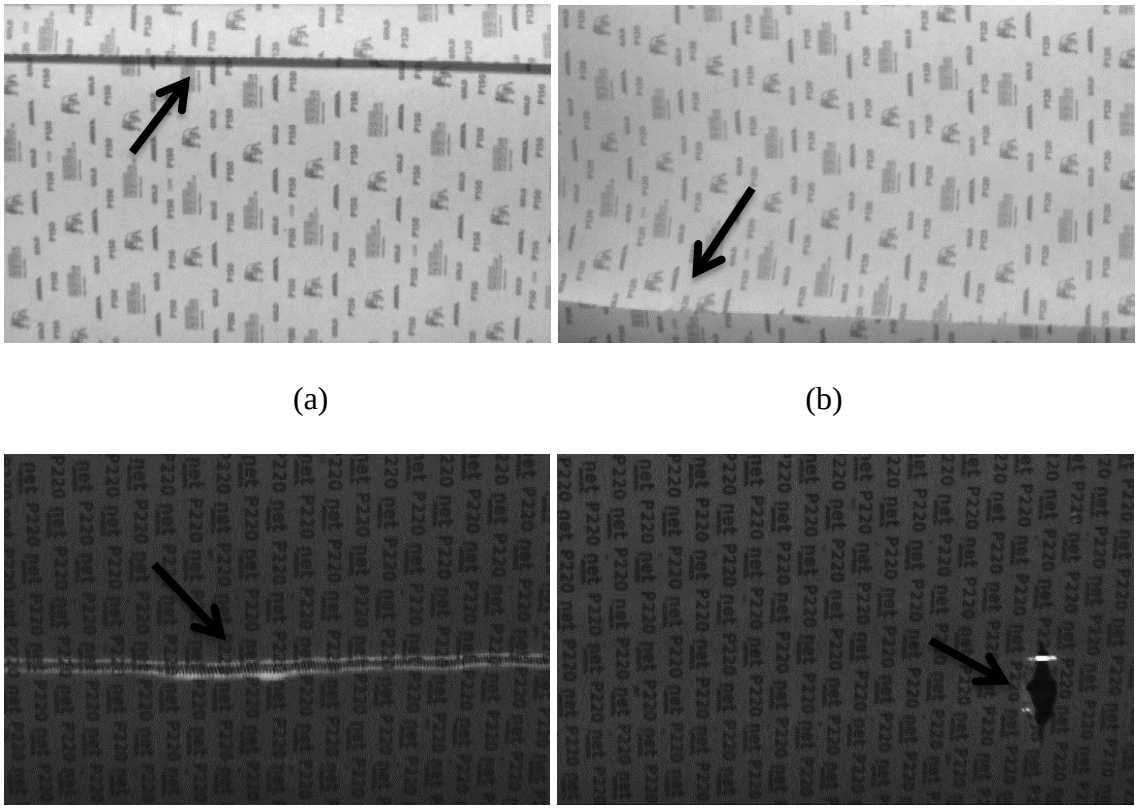
The projection of the feature vectors trough the membership functions is taken care of by a function named eval_membership(). The function takes as input a single feature vector, the parameters for the membership functions, and the value of plotON.

The function loops through all features and maps them through the membership functions for both materials. The first feature is mapped trough a membership function that assigns a value of 1.0 if the value is equal to the midpoint, and 0.7 otherwise. The other values are mapped trough the Gaussian functions described by the input parameters (see equation 21 in section 2.4.5). Remember that the function midpoint is saved in a variable named alpha, and that α_l and α_r in equation 21 are equal to alpha – midL and alpha + midH respectively. After this the values are combined into a single value by taking their product. In order to give extra weight to the minimum value, it is included twice in the product. The membership values calculated this way become quite low, since we have as many as 8 different features. If, for example, all features got a membership value of 0.9 the total combined membership would be 0.39.

5. TESTING AND RESULTS

5.1. The FZ5-system

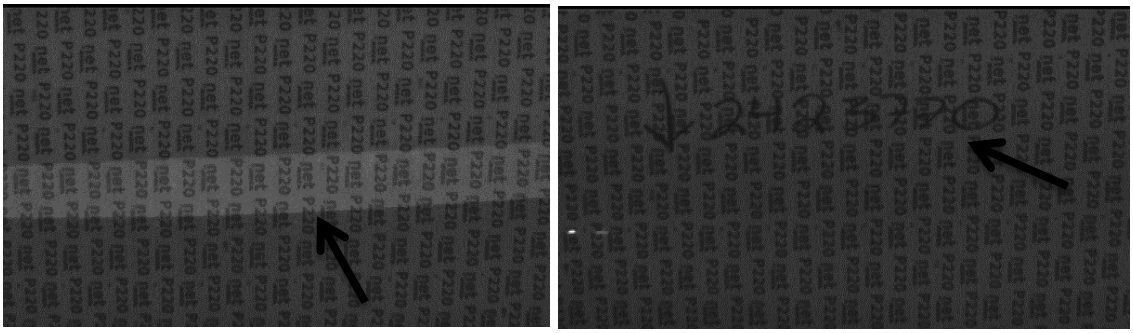
To begin this project, images where first logged with a single camera and used to perform offline tests. Images of only Autonet and Gold material were logged. Figures 27 – 30 shows some images with defects that the “Gravity and Area” and “Defect” methods were tested on in offline measurements. Parameters where set so that the system correctly judged all faultless images.



(a) (b)

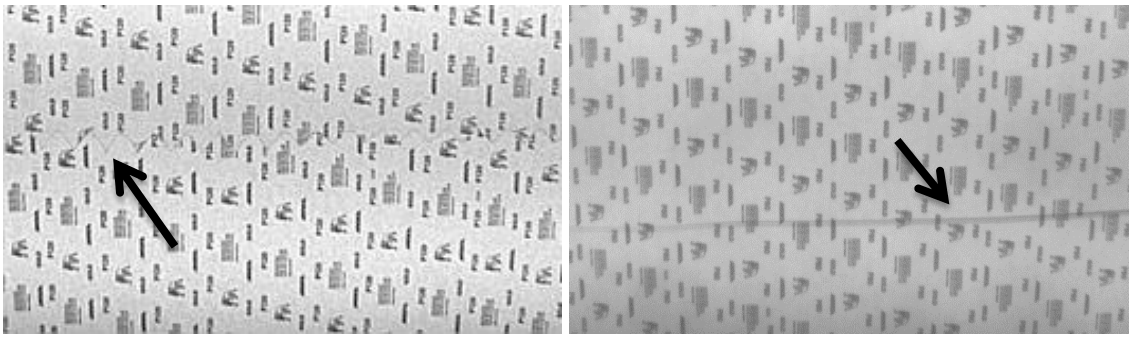
(c) (d)

Figure 27. These large area defects with deviant intensities were reliably detected by the Omron system in the offline tests. (a) a joint were the layers do not lie flat against each other casts a shadow visible as a dark band. (b) shows another similar joint as in (a). (c) the net-material have been joined by stitching. (d) a hole have been ripped in the material.



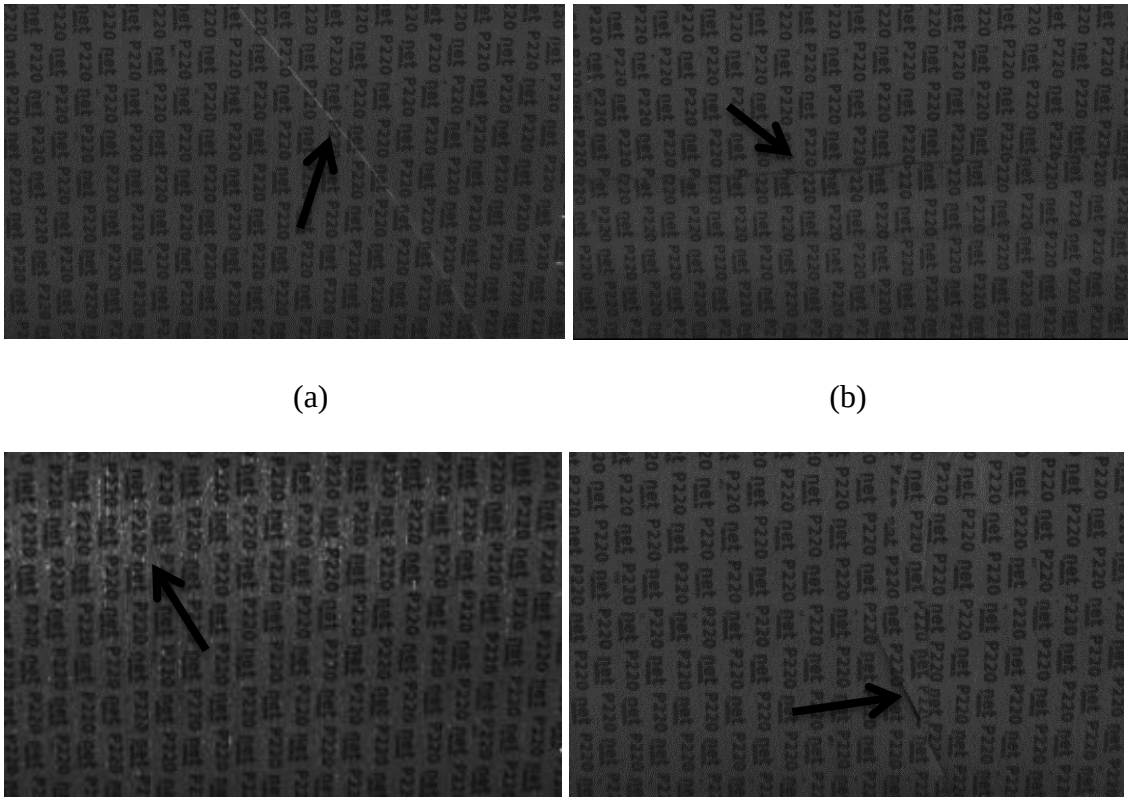
(a) (b)

Figure 28. These defects were detected in some images by the Omron system in off-fline tests (several images were captured of the same defect as it moved by). (a) a band with slightly higher intensity than normal. The cause of this band is unknown. (b) someone have used a marker pen to mark a specific spot.



(a) (b)

Figure 29. Defects in Gold material that could not be detected by the FZ5-system. (a) a glued joint with nearly identical intensities at both sides. After a bit of processing, a very thin zigzag line can be seen, but texture analysis of the prints would be required to detect the joint in software. (b) a thin line drawn by a marker pen. The line is too narrow, and has an intensity too close to the intensity of the printings, to be detected. Shape analysis would be required to detect this line.



(a) (b)

(c) (d)

Figure 30. Defects in Autonet that could not be detected by the FZ5 system. (a) a scratch visible as a bright line. (b) a horizontal fold. (c) these bright marks are caused by unsuccessful coating processes. Glue or some other chemical may have formed small local accumulations. Image processing has been used to enhance visibility. (d) another fold in the material.

When the results were satisfying, the online system was programmed and taken into use. Both the algorithm and parameters were now slightly adjusted, to match the two camera setup and to work for all materials. A two hour test was performed on Autonet. The machine was not stopped due to a single false alarm. In that time, two different defects were detected, a joint similar to the one in Figure 27 (c) and a hole similar to the one in Figure 27 (d). At least one defect passed by undetected, a small spot with a size of about 1 cm^2.

A several hours long test was done also on Gold material. During this time, several defects were detected, but the machine was also stopped due to false alarms a couple of

times. Parameters were adjusted to lower the sensitivity. Figure 31 shows an example of a typical case that causes problems. Figures 32 – 39 shows defects that were detected during the test. Check the list on the right side of the GUI to see which methods detected the defect. The icon of a method that detects a defect turns red.

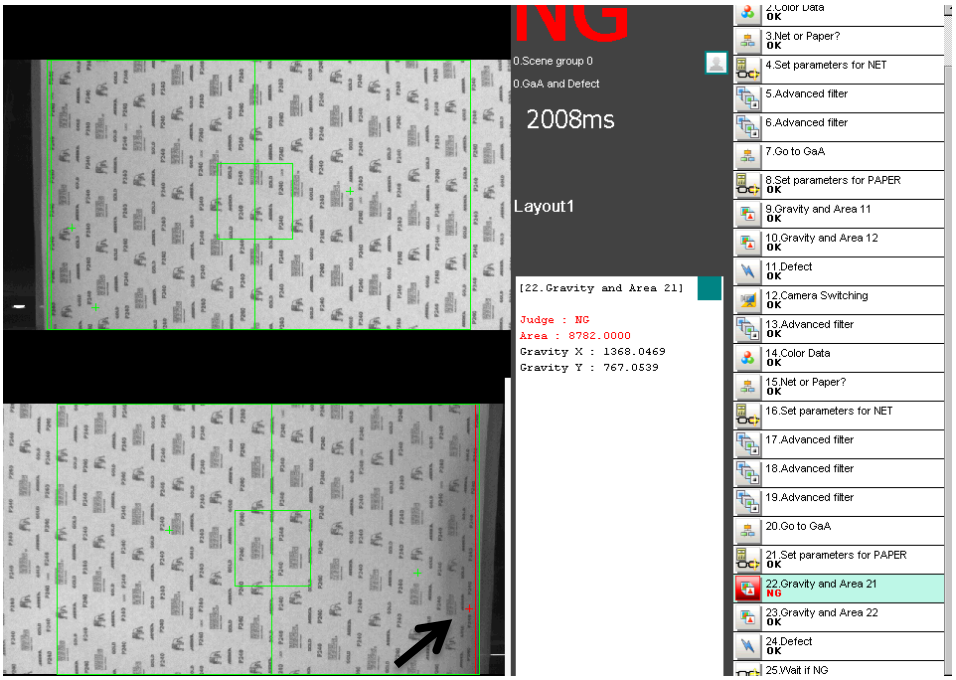


Figure 31. A dark shadow is formed as the material is slightly folded near the right edge. If this happens frequently, the sensitivity must be lowered.

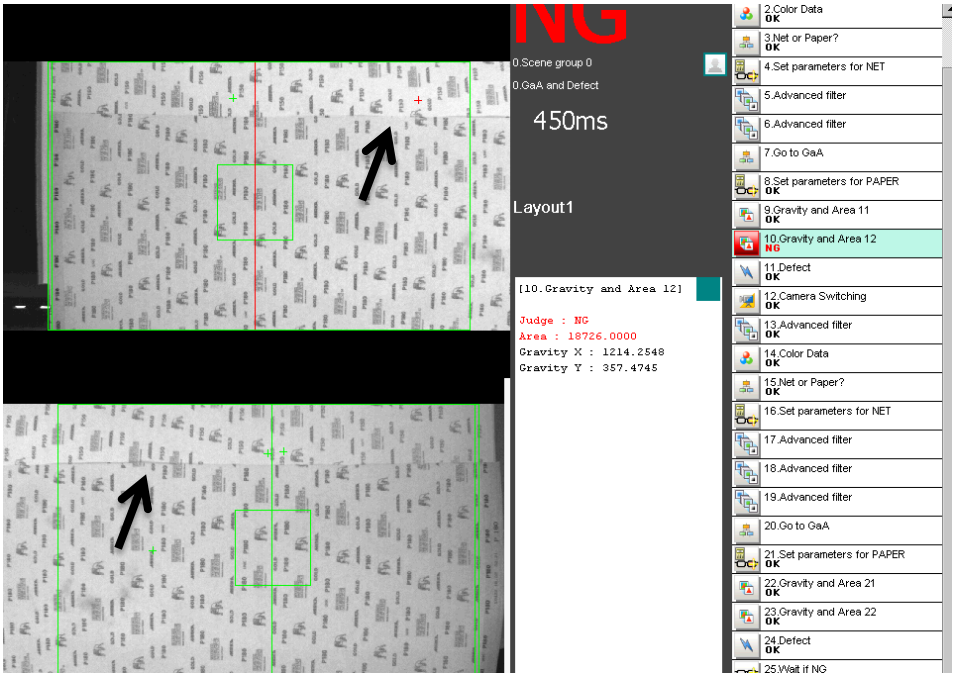


Figure 32. A joint is detected, but only by one of the six methods.

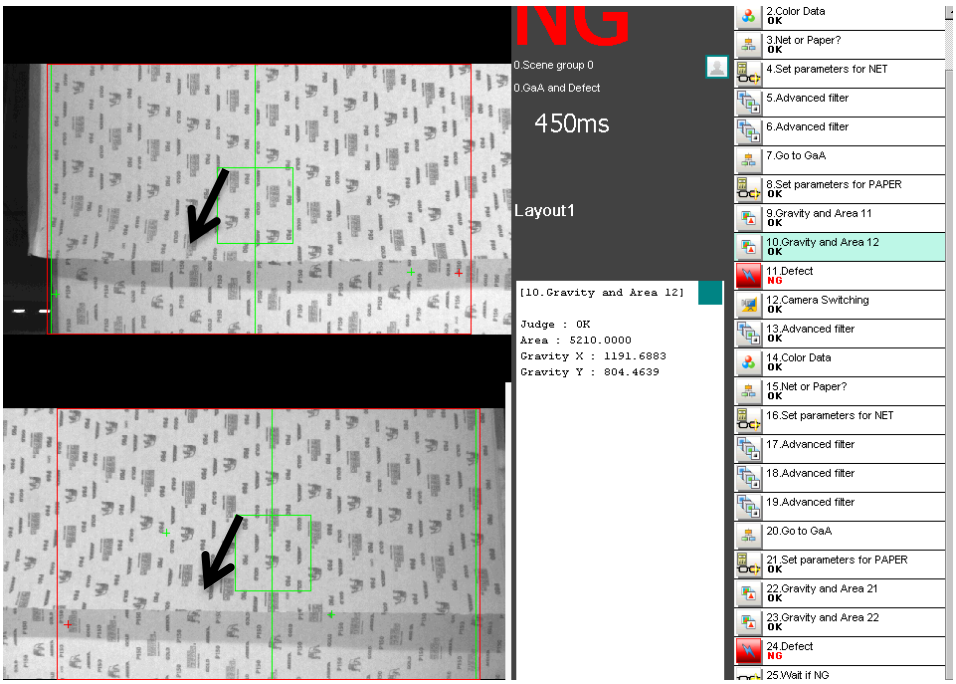


Figure 33. A different kind of joint, detected by the “Defect” method in both images.

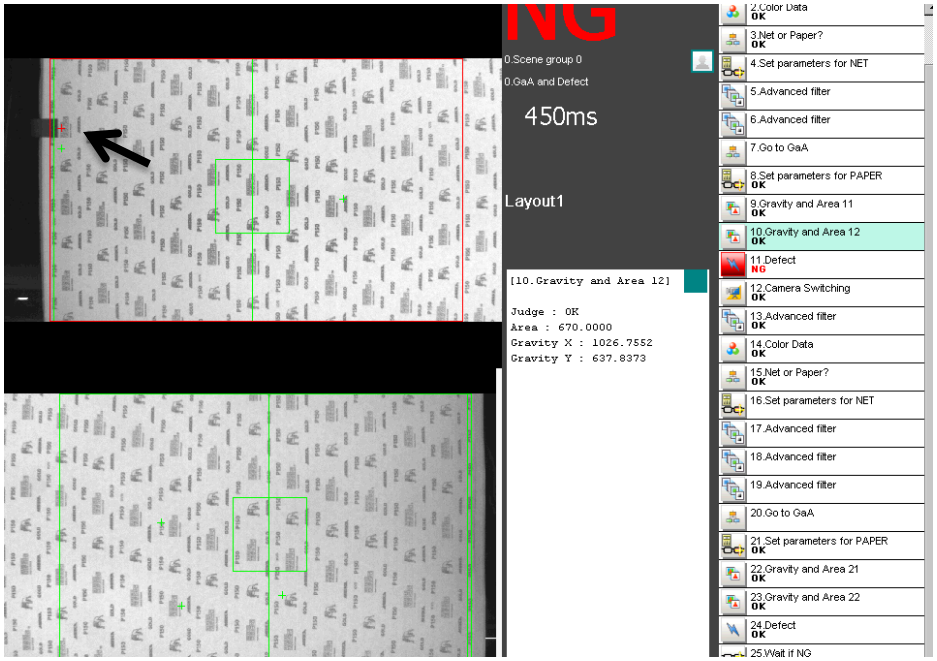


Figure 34. The "Defect" method detects a flag in the left edge of the material. Flags are used to mark defects that have been detected in earlier stages.

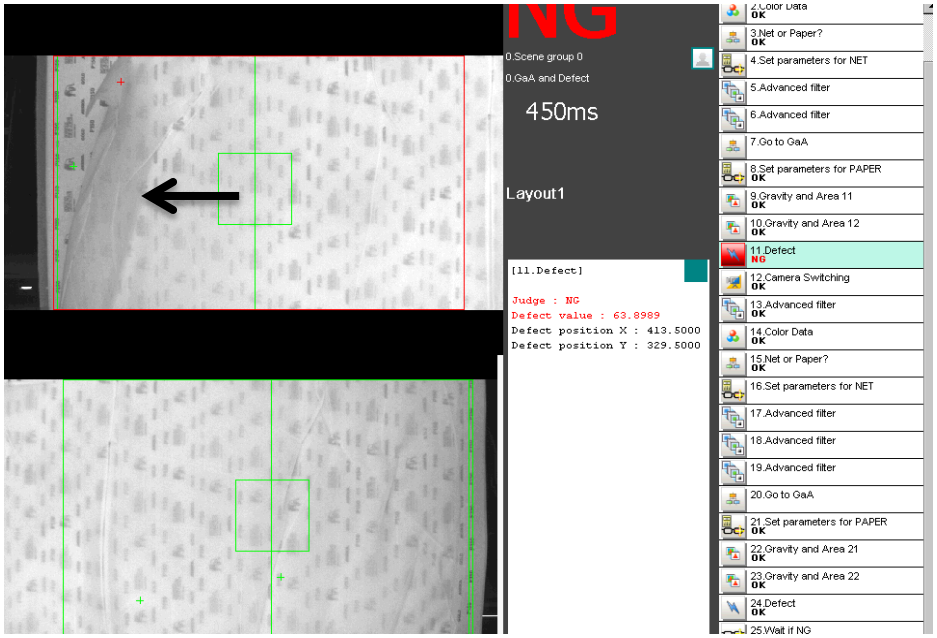


Figure 35. Loose grip cloth is detected by the "Defect" method when a dark shadow is formed near the left edge. Uniformly coloured defects cannot be detected, since the defect covers entirely the second image, it is not detected there.

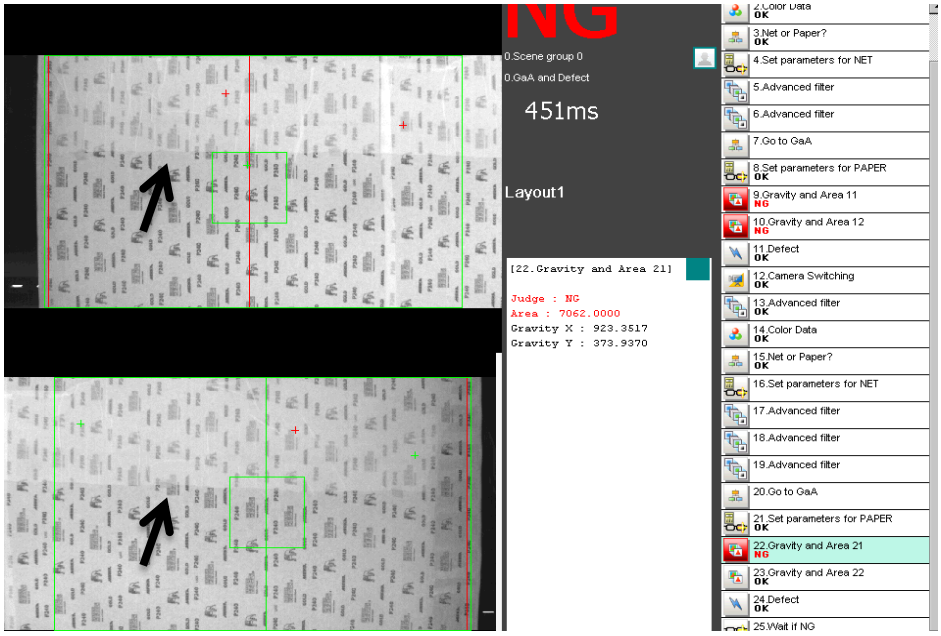


Figure 36. The edge of overlapping grip cloth is detected, since now there is a notable difference in intensities and the "Gravity and Area" methods reacts.

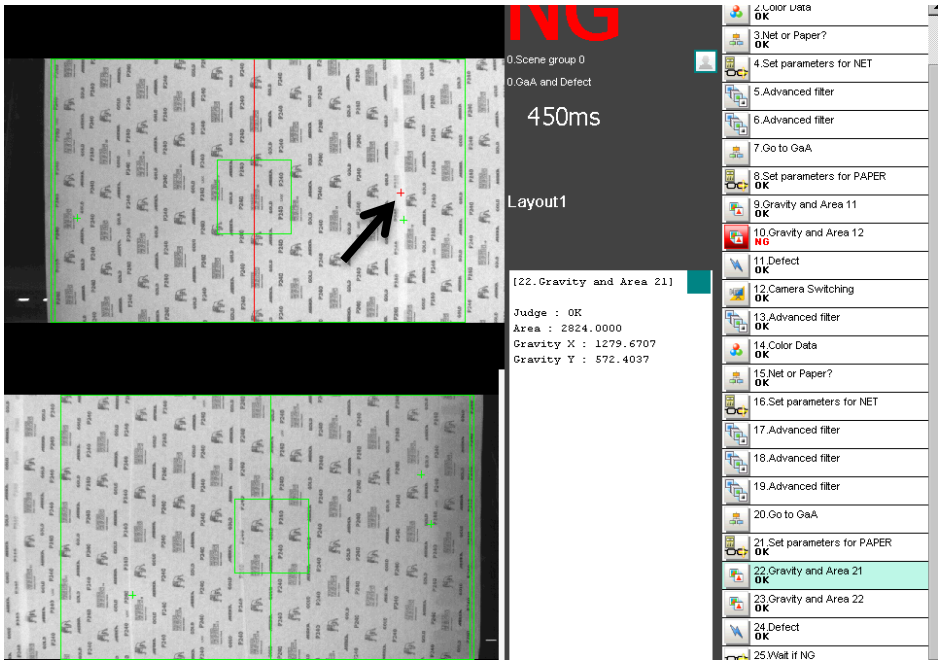


Figure 37. A bright band is detected in the left image by "Gravity and Area".

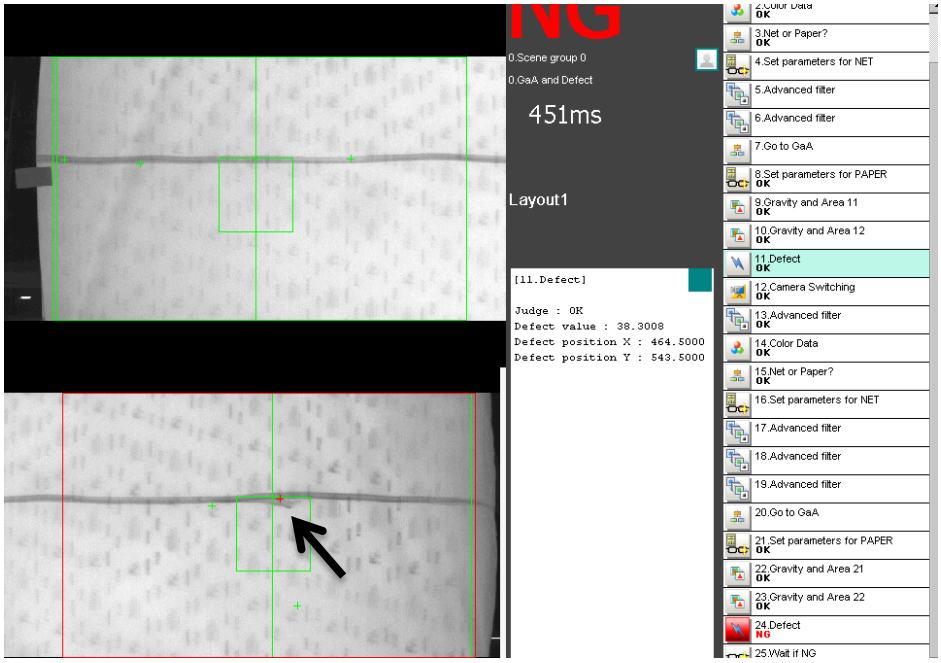


Figure 38. The width of this stripe is near the limit of what can be detected. It is not detected in the upper image, but in the bottom image it is slightly broader in a short section. At this location it is detected by the "Defect" method.

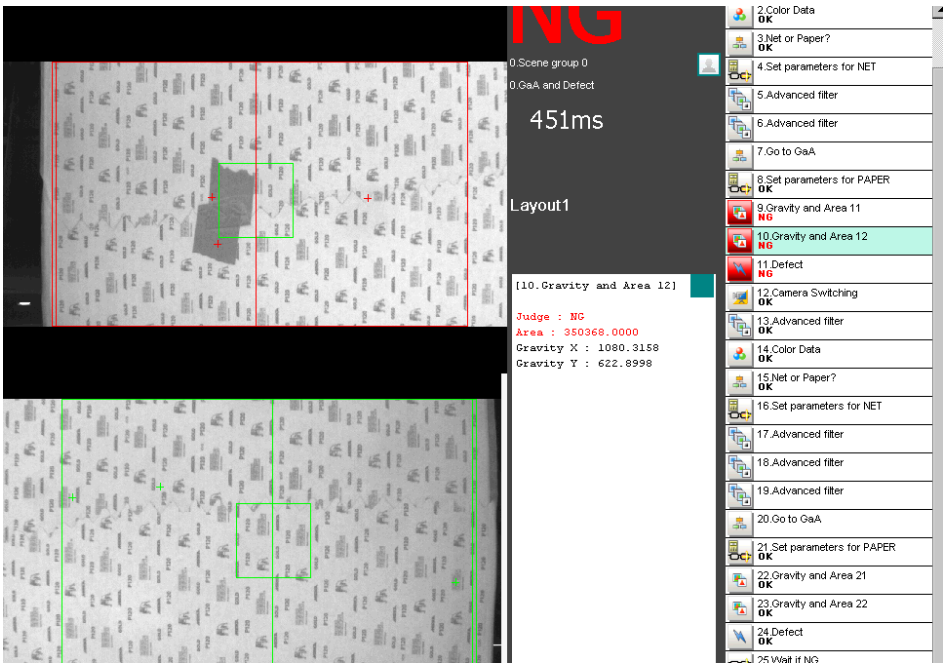
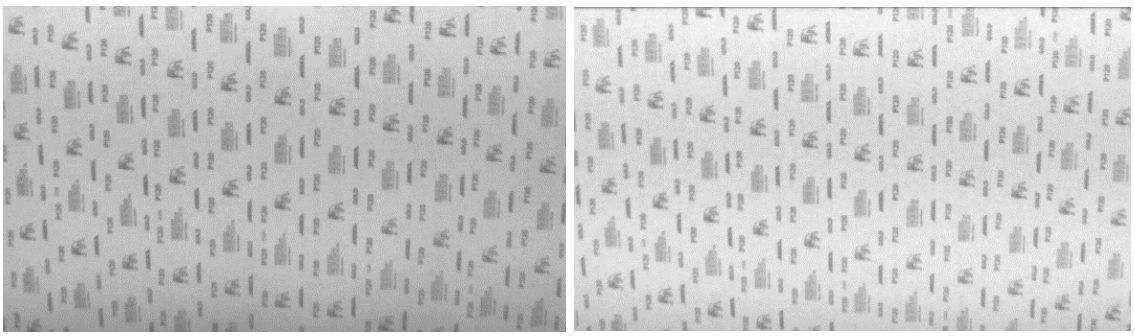


Figure 39. A joint has been strengthened with some tape, which is detected.

5.2. Texture analysis with the Fourier transform

Testing of this algorithm was done on the images logged of Gold and Autonet in the beginning of the project. It became clear that this algorithm also would require parameter adaption over time when it runs. Two images of Gold material logged at different days are shown in Figure 40. The majority of Gold images were logged during a single run lasting a couple of hours. Five images alone were logged at another day to see if the results were consistent. They were not. One reason for this appeared to be the amount of dust on the lights. The algorithm should not be sensitive to the total light intensity, but it seems that the dust scatters the light so that the illumination becomes more diffuse and uniform.

The generated membership functions are shown in Figures 41 – 48. Feature one is not mapped trough a membership function, so the plots shows features 2 – 8 for both Gold and Autonet. Data points for Gold are shown as green crosses, while data points for Autonet are shown as blue circles. As discussed earlier, only the feature values for a single material are used at a time. The data points for the other material are included in the plots merely for visualization. The y-value of a data point is the membership assigned to the point by the fuzzy c-means clustering algorithm. Note that for Gold, there are five points that have been assigned significantly lower values than the rest. For Autonet, there are also a couple of points that cause problems. A *k*-value of 10 is used.



(a) (b)

Figure 40. Images of Gold logged at different days. The environment is quite dirty and the difference in illumination between the two images is probably due to the amount of dust on the line lights.

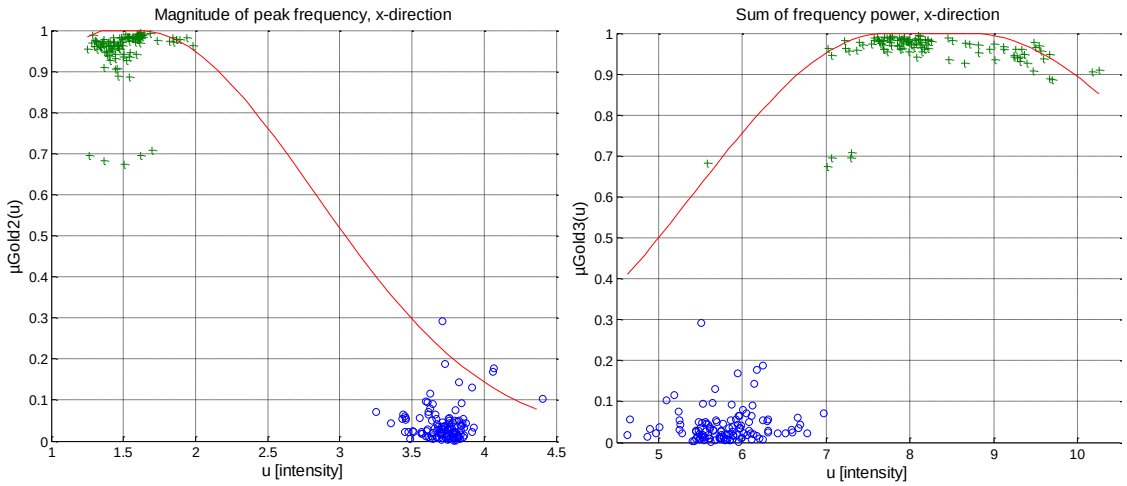


Figure 41. Membership functions for feature 2, magnitude of peak frequency in x-direction, and feature 3, sum of frequency power in x-direction, for Gold.

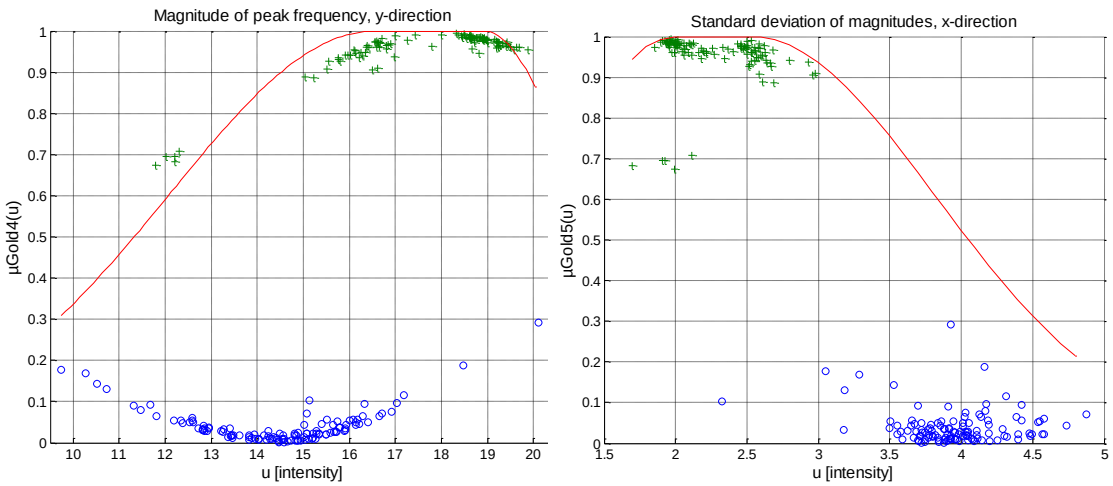


Figure 42. Membership functions for feature 4, magnitude of peak frequency in y-direction, and feature 5, standard deviation of frequency power in x-direction, for Gold.

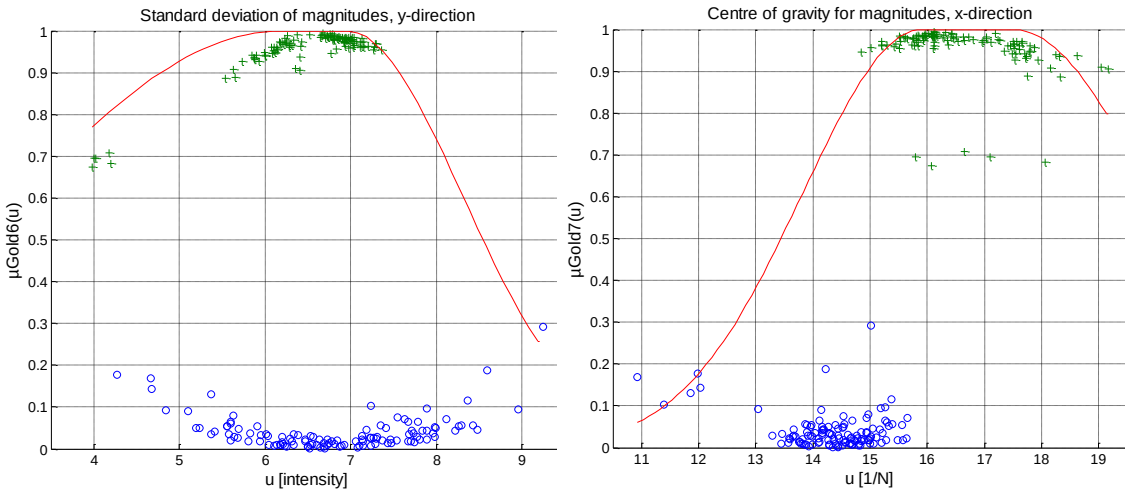


Figure 43. Membership functions for feature 6, standard deviation of frequency power in y-direction, and feature 7, centre of gravity for frequency power in x-direction, for Gold.

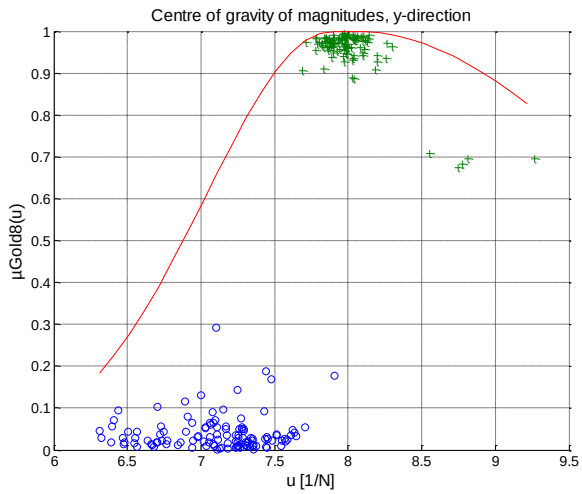


Figure 44. Membership function for feature 8, centre of gravity for frequency power in y-direction, for Gold.

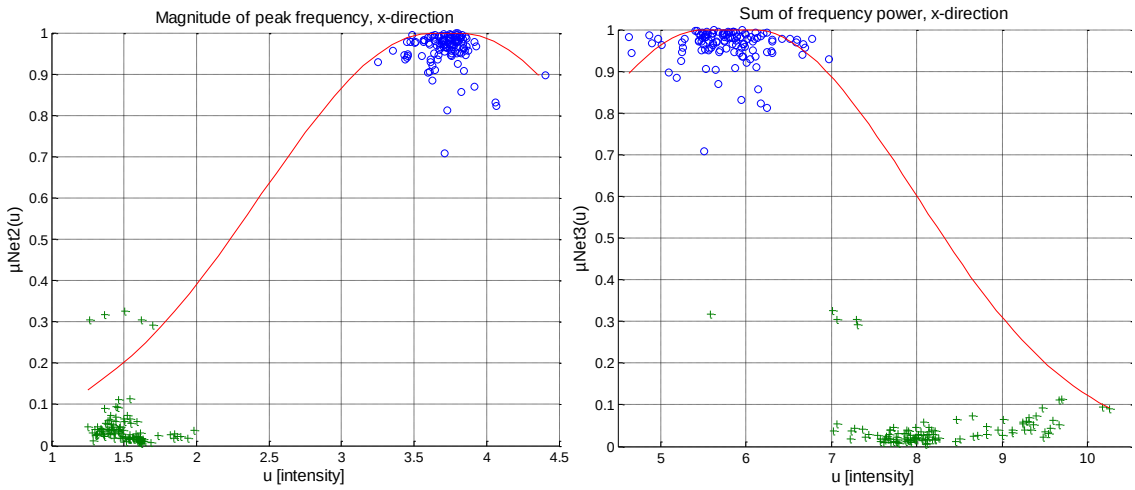


Figure 45. Membership functions for feature 2, magnitude of peak frequency in x-direction, and feature 3, sum of frequency power in x-direction, for Autonet.

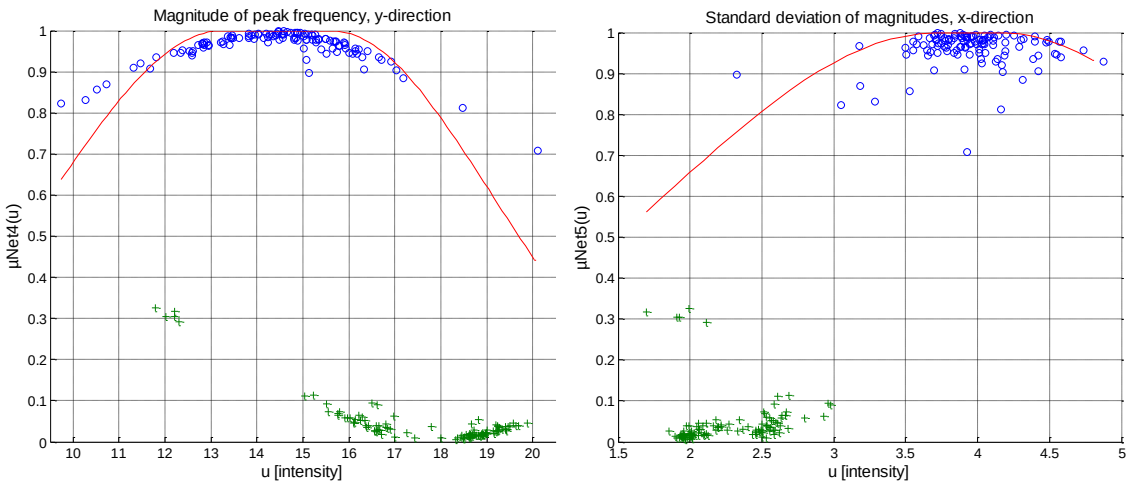


Figure 46. Membership functions for feature 4, magnitude of peak frequency in y-direction, and feature 5, standard deviation of frequency power in x-direction, for Autonet.

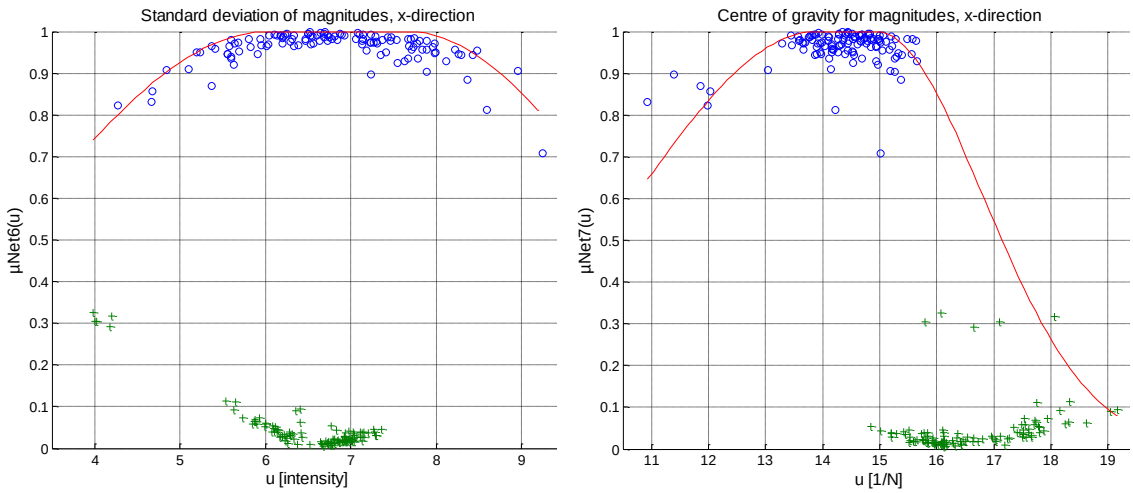


Figure 47. Membership functions for feature 6, standard deviation of frequency power in y-direction, and feature 7, centre of gravity for frequency power in x-direction, for Autonet.

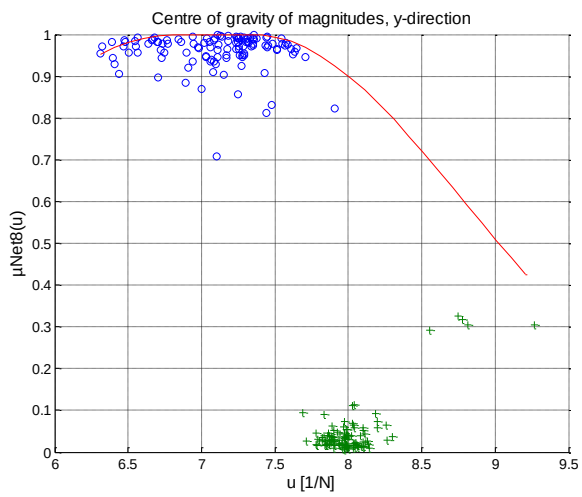


Figure 48. Membership function for feature 8, centre of gravity for frequency power in y-direction, for Autonet.

A large *k*-value, in this case 10, is needed to generate membership functions that assign acceptable memberships to all data points, see for example feature 4 and 6 for Gold. However, this large a value makes the algorithm quite insensitive, and in an online application where we do not analyse images taken at different days at the same time, a smaller value would give better performance. However, in the following test we keep the *k*-value of 10.

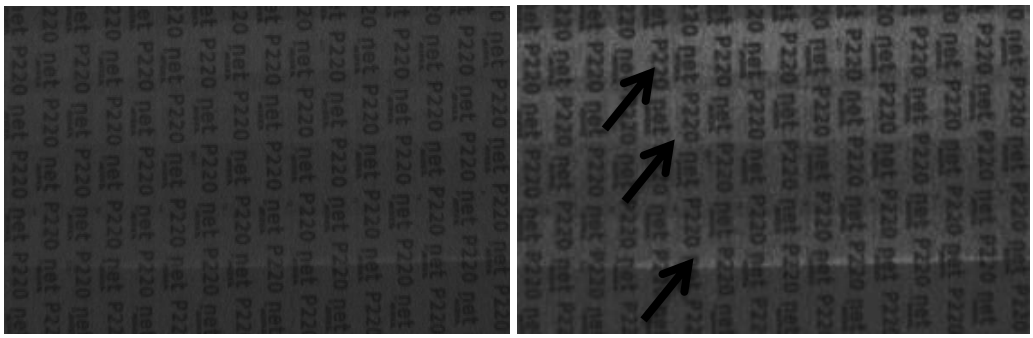
The generated parameters are first used to analyse a set of faultless images. The total number of faultless images analysed is 108 images of Gold and 118 of Autonet. The ten lowest memberships for both materials are listed in Table 2. To correctly judge all images, quite low limit values would have to be used.

Table 2. Lowest memberships for faultless images.

Gold	0.19	0.20	0.21	0.24	0.31	0.49	0.52	0.63	0.79	0.79
Autonet	0.13	0.20	0.21	0.32	0.37	0.37	0.40	0.62	0.67	0.69

The same parameters are then used to analyse a set of images that contain defects. The limit value is set to 0.3 for both materials, which would have caused some faultless images to be misjudged. However, the attention should be on the membership values and not on the final conclusion. What follows is a summary of the results. A series of images used in this test, along with results from the algorithm, can be found in Appendix D.

Large area intensity changes are detected, such as holes and tape joints. Even small changes in intensity are enough as long as the area is large enough to have impact on the magnitude spectrum. An example is shown in Figure 49.



(a) (b)

Figure 49. This image contains three bands that are just slightly brighter than normal and barely visible to the eye. Area is large enough to be detected. (a) original image used in the test. (b) the same image that have been processed for better visibility of the defects.

Small area intensity changes are usually not detected, even if the defect is significantly brighter or darker than the surrounding material. The total number of pixels in the image is quite large, so if only a few differs from normal, the impact on the spectrum is unnoticeable. An example is shown in Figure 50.

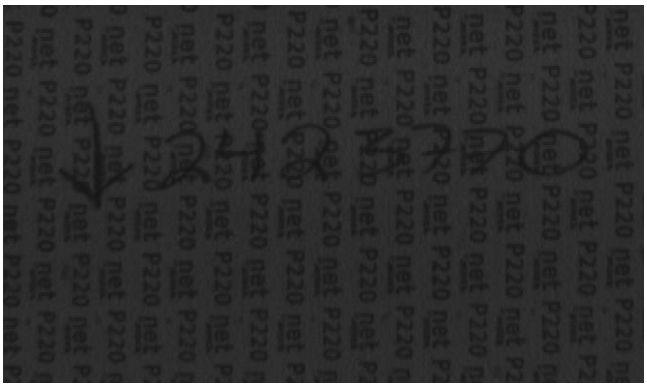


Figure 50. The total area of this defect is quite small, and the algorithm has problems detecting it, even though the intensity values are significantly lower than in the rest of the image.

Folds that changes positions of prints are reliably detected in at least the Autonet material. An example of this kind of defect was shown in Figure 21. Some images of Gold material with these kind of defect were also analysed, but only very few. Every one of these were detected, but they all contained also abnormal intensity values, and it is therefore hard to tell the cause of detection.

Scratches in the material are occasionally detected if their relative positions happen to cause a strong response to some particular frequency, but usually they do not. An example of scratches in Autonet is shown in Figure 51.

Figure 51. Example of scratches in Autonet.

5.3. Summary of the results

The FZ5-system detects intensity variations reliably, as long as the difference in intensity is large enough compared to the background. Some false alarms are caused by shadows near the edges, especially on paper-type materials. Better detection capabilities and more stable performance could be achieved by using other available methods in the library, but this would require a system version with more processing power. Changes in texture cannot be detected by the available methods.

The algorithm developed in Matlab reacts strongly to changes in relative positions of prints. It detects folds and joints that shift the prints, and also large area intensity changes. Defects that differ significantly in brightness are not necessarily detected if the area is too small.

6. CONCLUSIONS

A complete machine vision system, based on the Omron FZ5, has been implemented for defect detection in KWH Mirka's production line. It fulfils the minimum requirements listed in Section 1.1. The algorithm reacts only to intensity variations, since there were no methods available in the library that was well suited for texture analysis. If improvements are to be done, the first thing to do would be to improve the lighting near the edges of the material. Getting more uniform lighting would open up for a more sensitive system without the risk of stops due to misclassification of faultless material. Investing in a high-speed version of the FZ-controller would give better results, but it would probably only be a minor and expensive improvement.

The Omron FZ5-system is not suitable at all to detect changes in texture. If sensitivity to defects that do not cause intensity variations is needed, another platform must be used. A program was developed in Matlab to test whether it is possible to use the Fourier transform to analyse the printings on the backside of the abrasive materials. The method was able to detect shifts in the relative positions of printings, which usually indicate folds or a joint. This method relies on parameters extracted from model images. A function was written that automatically calculates these parameters, and in an online version, this function should be used to continuously adapt them.

6.1. Future work

More research on the effects of different defects on the frequency spectrum would open up for optimizing the features that are used. Correct choice of features is the key to successful detection. Limiting the use of the spectrum only to the x- and y-axes was a way to simplify the selection, but it would be interesting to analyse also other regions of the spectrum. Research on the behaviour of the FFT itself could also be beneficial, for example how border effects affect the result, and if the use of window functions could improve the result. An implementation that performs the Fourier Transform over several smaller areas would also be interesting to experiment with.

A combination of both texture and intensity analysis would probably yield the best results for an online system. This requires replacing the FZ5-system with a more powerful platform. I would use two line scan cameras with relatively low resolution, along with a high quality, focused line illumination. The flexibility of a standard PC would be desirable. Measuring and plotting the results for one image in the Matlab algorithm was done in about half a second on my computer with an Intel Core i5-3337U 1,80 GHz processor and Windows 8.1 operating system. Parallel processing or pipelining would be needed to implement both intensity and texture analysis along with parameter updates in online measurements. This could be realized by using a DSP or an FPGA located on a frame grabber board to implement the FFT, before sending the transformed images to the CPU that would handle feature calculations, mapping through membership functions and parameter updates.

REFERENCES

Aziz, Mahmoud Abdel; Ali S. Haggag; Mohammed S. Sayed (2013). *Fabric defect detection algorithm using morphological processing and DCT.* 2013 First International Conference on Communications Signal Processing and Their Applications.

Chan, Chi-ho; Grantham K. H. Pang (2000). *Fabric defect detection by Fourier analysis*. IEEE Transactions on Industry Applications, Vol. 36, No. 5. 1267 – 1276.

Koljonen, Janne (2013). *AUTO3110, Luento 5*. [Lecture slides from course AUTO3110 Machine vision]. Facoulty of technology, University of Vaasa

Litwiller, Dave (2005). *CMOS vs. CCD: Maturing Technologies, Maturing Markets.* Photonics Spectra, August 2005. Laurin publishing.

Martin, Daryl (2007). *A Practical Guide to Machine Vision Lighting*. White paper, Advanced Illumination.

Microscan (2013). *Eight tips for optimal machine vision lightning*. White paper, Microscan systems.

Microscan (2014). *Shedding light on machine vision*. White paper, Microscan systems.

Niederjohann, Britta (2014). *Fundamentals of Image Processing Sytems.* White paper, Basler AG.

Omron (2016). *Vision Sensor-FH/FZ5 Series Vision System Processing Item Function Reference Manual.*

Omron (2016). *Vision Sensor-FH/FZ5 Series Vision System User's Manual.* Page 70.

Omron (2014). *Quality Control & Inspection Guide 2014.* 72, 77 – 80, 84 – 89.

Oppenheim, Alan; Ronald Schafer; John Buck (1999). *Discrete-Time Signal Processing.* Prentice Hall, 2nd Edition. Chapter 9.

Pinter, Matt. *Strobe; a new era for LED's and machine vision lighting*. Articles and Application notes, Smart Vision Lights.

Rashidian, Behnam; Eric Fox (2011). *The Evolution of CMOS Imaging Technology*. White paper, Teledyne DALSA.

Schwär, Michael; Daniel Toth (2014). *Camera selection – How can I find the right camera for my image processing system?* White paper, Basler AG.

Steger, Carsten; Markus Ulrich, Christian Wiedemann (2008). *Machine Vision Algorithms and Applications*. WILEY-VCH Verlag GmbH & Co. KgaA, 1st Edition, 4th Reprint. 89 – 93, 125 – 144.

Vision Light Tech (2014). *LED light source unit PFBR-150SW series*. Product Brochure, Vision Light Tech.

Von Fintel, René (2013). *Comparison of the Most Common Digital Interface Technologies in Vision Technology*. White paper, Basler AG.

Yang, X.; G. Pang; Yung N. (2005). *Robust fabric defect detection and classification using multiple adaptive wavelets*. IEE Proceedings –Vision Image and Signal Processing, Vol. 152, No. 6, December 2005. 715 – 723.

Zuo, Haiqin; Yujie Wang; Xuezhi Yang; Xin Wang (2012). *Fabric defect detection based on texture enhancement.* 5th International Congress on Image and Signal Processing. 876 – 880.

APPENDIX A

Figure 1A shows the blocks used in the Omron FZ program. The figure is a print screen of the flow viewer window.

0. Camera Image Input. This block communicates with the connected cameras. Used to set camera parameters and acquire images from both cameras.
1. Advanced filter. This block can apply several sequential filter operations to an image. Used for dilating the image.
2. Color Data. Calculates mean intensity and standard deviation when used on monochrome images. Used to classify materials based on the intensity value.
3. Conditional Branch. Used to make the program jump to specific locations depending on the result of a Boolean expression.
4. Calculation Macro. Can be used to write several macros that output numerical values. This block has a graphical window where one can set limits for the results of the implemented calculations and thus get an output judge of “OK” or “NG”. This instance of the block is used to calculate and set parameters for the “Gravity and Area” and “Defect” methods if the material is of net-type.
5. Advanced Filter. Used to remove a single bright spot if the material is of net-type. Only one ROI can be specified, so several blocks are needed to remove several spots.
6. Advanced Filter. Used to remove a bright spot.
7. Conditional Branch. Skips block number 8 if material is of net-type. (The programing of the system is performed by dragging every block into a single list)
8. Calculation Macro. Calculates and sets parameters for the “Gravity and Area” and “Defect” methods if the material is of paper-type.
9. Gravity and Area. Measures one half of the input image.
10. Gravity and Area. Measures the other half of the input image.
11. Defect. Measures the input image for local intensity variations.
12. Camera switching. Used to change the measurement image when several images have been acquired into memory. Used to change to the second camera image.

13 – 24. Perform the same steps on the second camera image.

25. Macro. Block for writing program macros in a text editor. Used to check the total judgement result and perform a wait instruction if the result is “NG”.

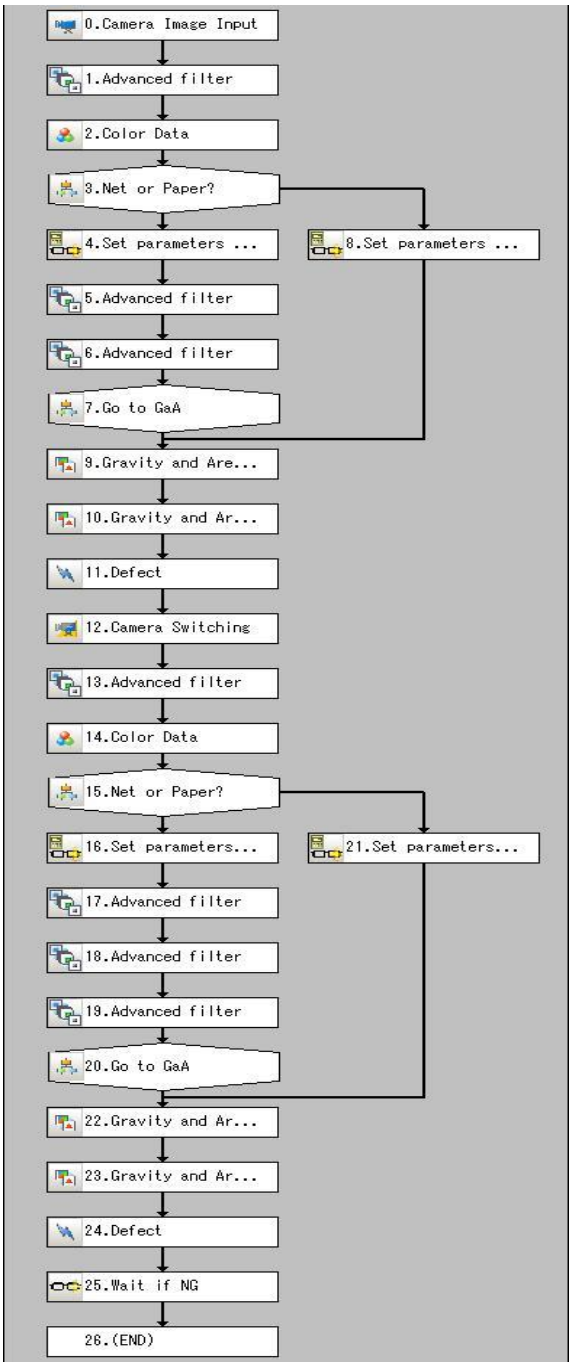


Figure 1A. The flow of the FZ program as viewed in the flow viewer window.

APPENDIX B

The Matlab code for the function that calculates the parameters for the fuzzy membership functions. A detailed description can be found in chapter 4.

```
function [ params ] = abra_fuzzyParams( Nrules, k, plotON )
%This is the function for generating parameters for the fuzzy
%membership functions used to identify and check abrasive materials for
%defects. The parameters can be used as input to the function meas-
ure_abrasives()
%which uses them for measurements.

%The images that are used as data to generate parameters must be located in
% the directory specified by path.
% Nrules is the number of rules to be generated, which should be the same as
the number
% of different materials to evaluate.
% if plotON is not zero, then the membership functions along with data points
are plotted as
% they are generated
    pause on;
    path = 'C:\DItyo\Allt-OK\';

    % Get list of image filenames
    imageList = getAllFiles(path);
    % Evaluate their features
    FM = abra_features(imageList);
    % Cluster the feature matrix into Nrules clusters
    %parameters
    param.c=Nrules;
    param.m=2;
    param.e=1e-6;
    clust = fuzzy_clustering(FM, param);
    % Output cluster midpoints
    params.alpha = clust.v;
    % Calculate beta values and midpoint offsets
    [betas, midH, midL] = calc_betas(FM, clust, k, plotON);
    % Output values
    params.beta = betas;
    params.midH = midH;
    params.midL = midL;
    params.Nrules = Nrules;
    pause off;
end

function FM = abra_features( imageList )
%% This function reads the image files in imageList, and
% analyses their frequency content. The results are returned in a feature
% matrix FM

% The program extracts the following features:
% The index and magnitude of strongest peak in the x-direction on the inter-
vall [1;50] Hz
% The sum of all frequency components from 1 Hz to the peak index
% The sum of all frequency components in the y-direction on the intervall
[1;27] Hz
% Standard deviation in both x and y directions, on the same intervalls
```

```
% Center of gravity in both x and y directions, on the same intervalls

    %Initialize feature matrix
    FM = zeros(length(imageList), 8);

    % Loop through images
    for i = 1:length(imageList)
        %% Preprocessing
        %Read and crop input image
        imName = char(imageList(i));
        %imName = strcat(path, imName);
        img = imread(imName,'bmp');
        img = img(559:1079,370:1290);

        %Perform trapezoidal correction
        %image x midpoint
        mx = length(img(1,:))/2;
        temp = img;

        for y = 1:length(img(:,1))
            for x = 1:length(img(1,:));

                dx = round(32*(((mx-x)/mx)*(y/526)));
                temp(y,x) = img(y,x+dx);
            end
        end

        img = temp;

        %Filter the input image
        H = fspecial('gaussian',5);
        img = imfilter(img,H);
        %%
        % Calculate features
        FM(i,1:8) = freq_features(img);
    end
end

function outputVector = freq_features( img )

    % Initialize output vector
    outputVector = zeros(1,8);
    [x y] = size(img);
    % Calculate fft magnitude
    freqImg = abs(fft2(img)/(x*y));

    % normalize to DC component
    freqImg = freqImg/freqImg(1,1);

    % Get the max value and corresponding index along x direction
    [MagX,IndX] = max(freqImg(1,2:50));
    % Sum of all values up to this index
    sumX = sum(freqImg(1,2:IndX));
    % Sum of all frequency components up to 3*9 = 27Hz in the y direction
    sumY = sum(freqImg(2:27,1));
    % Standard deviation in the x direction
    if IndX>1
        stdX = std(freqImg(1,2:IndX));
    else
        stdX = 0;
    end
    % Standard deviation in the y direction
    stdY = std(freqImg(2:27,1));
    % Center of gravity in x direction
```

```
    if IndX>1
        tx = 1:IndX-1;
        CoGX = freqImg(1,2:IndX)*tx'/sumX;
    else
        CoGX = 0;
    end
    % Center of gravity in y direction
    ty = 1:26;
    CoGY = freqImg(2:27,1)'*ty'/sumY;

    % Set outputs
    outputVector(1,1) = IndX;
    outputVector(1,2) = MagX*100;
    outputVector(1,3) = sumX*100;
    outputVector(1,4) = sumY*100;
    outputVector(1,5) = stdX*1000;
    outputVector(1,6) = stdY*1000;
    outputVector(1,7) = CoGX;
    outputVector(1,8) = CoGY;
end

function fileList = getAllFiles(dirName)

    % Get the data for the current directory
    dirData = dir(dirName);
    % Find the index for directories
    dirIndex = [dirData.isdir];
    % Get a list of the files
    j=1;
    for i = 1:length(dirData(:,1))-1
        if ~dirIndex(i)
            fileList(j,:) = cellstr(dirData(i).name);
            j=j+1;
        end
    end
end

function result = fuzzy_clustering( FM, clustparams )
%% An implementation of the fuzzy C-means clustering algorithm
    %% Algorithm Init
    % Set parameters
    c = clustparams.c;
    if exist('clustparams.m') == 1, m = clustparams.m; else m = 2; end
    if exist ('clusterparams.e') == 1, e = clustparams.e; else e = 1e-4; end

    % Init fuzzy partition matrix
    U = rand(length(FM(:,1)),c);
    U = U./(diag(sum(U,2))*ones(size(U)));

    Uold = U+1;
    v = zeros(c,length(FM(1,:)));

    %% Iterate
    while max(max(abs(U-Uold))) > e
        %% Calculate midpoints
        Um = U.^m;
        for cn = 1:c
            for i = 1:length(FM(1,:))
                v(cn,i) = sum(FM(:,i).*Um(:,cn))/sum(Um(:,cn));
            end
        end

        %% Update memberships
        Uold = U;
```

```
        % Data point distances to midpoints
        for i = 1:c
            norm_x_v(:,i)=sqrt(sum((FM - ones(size(FM))*diag(v(i,:))).^2,2));
        end

        % Check if data points coincide with any midpoint
        minD = min(min(norm_x_v));
        if minD == 0
            % then find these indicies.
            I = find(norm_x_v == 0);
            % Change these distance values so that we do not get division by
            % zero when calculating memberships. Correct memberships for
            % these data points are set later
            norm_x_v(I) = 0.001;
        end
        % Calculate memberships
        for i = 1:c
            temp = 0;
            for j = 1:c
                temp = temp + (norm_x_v(:,i)./norm_x_v(:,j)).^(2/(m-1));
            end
            U(:,i) = 1./temp;
        end

        %Set correct memberships for points that coincide with midpoints
        if minD == 0
            [row col] = find(norm_x_v == 0);
            U(row,:) = 0;
            U(I) = 1;
        end
    end

    result.v = v;
    result.U = U;

end

function [betas, midH, midL] = calc_betas( FM, clust, k, plotON )
% This function calculate midpoint offsets and beta values. The midInterval
% value specifies how many percent of the data points that should be included
on
% the intervall of constant one. Feature 1 is not mapped trough a membership
function
%
% FM = Feature matrix
% clust = the result from the fuzzy clustering algorithm (contains feature
midpoints)
% k = the constant in B=1/(2ks)
% plotON = membership functions along with data points are plotted as
% they are generated if this parameter is not set to 0
%
% betas = beta values for membership functions
% midH = midpoint "high-side" offsets
% midL = midpoint "low-side" offsets

    midInterval = 0.75;

    % Loop trough all clusters
    for c = 1:length(clust.U(1,:))

        % Loop through all of the features in FM
        for dim = 1:length(FM(1,:))
```

```
    % Feature 1 is not mapped with gaussian membership function
    if dim == 1
        betas.high(c,1) = 0;
        betas.low(c,1) = 0;
    else
        %% Get all values outside the mid interval
        devPoints=zeros(2,round(length(FM(:,1))*(1-midInterval)/2));

        for i = 1:length(FM(:,1));
            % Check that the point is member "enough" in this cluster
            if clust.U(i,c) > 0.5
                % Get distance to midpoint
                dist = abs(FM(i,dim)-clust.v(c,dim));

                % Check if larger or smaller than the midpoint
                if FM(i,dim) > clust.v(c,dim)
                  % Check if the point belongs to the largest so far
                    [M I] = min(devPoints(1,:));
                    if M < dist
                        devPoints(1,I) = dist;
                    end
                else
                  % Check if the point belongs to the smallest so far
                    [M I] = min(devPoints(2,:));
                    if M < dist
                        devPoints(2,I) = dist;
                    end
                end
            end
        end

%% Calculate beta
% We generate 4 parameters, so that a flat
% region of constant 1 includes all values inside the mid interval.
% Then beta values are calculated from the standard
% deviation from the closest value in the interval

        % Get the values closest to alpha contained in devPoints
        mh = min(devPoints(1,:));
        ml = min(devPoints(2,:));
        % Calculate the standard deviation from these values
        % Distances
        devPoints(1,:) = devPoints(1,:) - mh;
        devPoints(2,:) = devPoints(2,:) - ml;
        % Square distances
        devPoints(1,:) = devPoints(1,:) .* devPoints(1,:);
        devPoints(2,:) = devPoints(2,:) .* devPoints(2,:);
        % Square root of the sum of squared distances divided by
        % number of samples
        s1 = sqrt(sum(devPoints(1,:))/length(devPoints(1,:)));
        s2 = sqrt(sum(devPoints(2,:))/length(devPoints(2,:)));

        % Beta = 1/(2ks)
        Bh = 1/(2*k*s1);
        Bl = 1/(2*k*s2);

        %%
        % Output values
        betas.high(c,dim) = Bh;
        betas.low(c,dim) = Bl;
        midH(c,dim) = mh;
        midL(c,dim) = ml;

        if plotON ~= 0
```

```
                    %% Plot result
                    x1 = min(FM(:,dim)):0.1:max(FM(:,dim));
                    gauss = x1;
                    for z=1:length(x1)
                       if x1(z)<clust.v(c,dim)-ml
                          gauss(z) = exp(-Bl*(x1(z)-(clust.v(c,dim)-ml))^2);
                        elseif x1(z)>clust.v(c,dim)+mh
                          gauss(z) = exp(-Bh*(x1(z)-(clust.v(c,dim)+mh))^2);
                        else
                            gauss(z) = 1;
                        end
                    end

                    x = FM(:,dim);
                    y = clust.U(:,c);
                    figure
                    scatter(x',y,10,round(y)+1);
                    hold all
                    plot(x1,gauss);
                    hold off
                    pause
                end
            end
        end
    end
end
```

APPENDIX C

The Matlab code for the function that checks images for defects with the help of the parameters extracted for the membership functions. A detailed description can be found in chapter 4.

```
function result = abra_measureImages(params, plotON)
%% Reads images in the folder specified by path and measures their membership
values
    %path = 'C:\DItyo\bitmaps-net\Defekter';
    path = 'C:\DItyo\Allt-OK\';
    memship_OK_limit = 0.3;
    % Get list of image filenames
    imageList = getAllFiles(path);
    %Initialize feature vector
    FV = zeros(1,8);
    pause on;

    % Loop through images
    for i = 1:length(imageList)

        %% Preprocessing
        %Read and crop input image
        imName = char(imageList(i));
        %imName = strcat(path, imName);
        img = imread(imName,'bmp');
        img = img(559:1079,370:1290);

        %Perform trapezoidal correction
        %image x midpoint
        mx = length(img(1,:))/2;
        temp = img;

        for y = 1:length(img(:,1))
            for x = 1:length(img(1,:));

                dx = round(32*(((mx-x)/mx)*(y/526)));
                temp(y,x) = img(y,x+dx);
            end
        end

        img = temp;

        % Measure time taken for extracting features and classifying (and
        % plotting!)
        time = cputime;
        %Filter the input image
        H = fspecial('gaussian',5);
        img = imfilter(img,H);

        %% Calculate features
        FV(1,1:8) = freq_features(img);

        %%
        % Calculate memberships
        [memship plothandle] = eval_membership(FV, params, plotON);
```

```
        etime = cputime-time;
        result.ms(i,:) = memship;
        result.t(i,1) = etime;
        if max(memship) > memship_OK_limit
            result.j(i,:) = 'OK';
        else
            result.j(i,:) = 'NG';
        end
        %%
        if plotON ~= 0
            % Show the original image (after trapezoidal correction)
            figure(2);
            imshow(temp);

            % Set titles
            title(plothandle(1),['Membership value Gold = ',
            num2str(round(memship(1)*100)/100)],'Fontsize',14);

            title(plothandle(8),['Membership value Autonet = ',
            num2str(round(memship(2)*100)/100)],'Fontsize',14);

            title(plothandle(7),['Time = ',num2str(etime),'s'],'Fontsize',14);

            for c = 1:length(memship)
                if memship(c) > memship_OK_limit
                  title(plothandle(c*7-4),'OK','Fontsize',18,'Color','green');
                else
                  title(plothandle(c*7-4),'NG','Fontsize',18,'Color','red');
                end
            end

            pause;
        end
    end
    pause off
end

function [ outputVector ] = freq_features( img )
%% This function extracts frequency features from the image img.
% The program extracts the following features:
% Image DC-component
% The index and magnitude of strongest peak in the x-direction on the inter-
vall [1;50] Hz
% The sum of all frequency components from 1 Hz to the peak index
% The sum of all frequency components in the y-direction on the intervall
[1;27] Hz
% Standard deviation in both x and y directions, on the same intervalls
% Center of gravity in both x and y directions, on the same intervalls

    % Initialize output vector
    outputVector = zeros(1,8);
    [x y] = size(img);
    % Calculate fft magnitude
    freqImg = abs(fft2(img)/(x*y));

    % normalize to DC component
    freqImg = freqImg/freqImg(1,1);

    % Get the max value and corresponding index along x direction
    [MagX,IndX] = max(freqImg(1,2:50));
    % Sum of all values up to this index
    sumX = sum(freqImg(1,2:IndX));
    % Sum of all frequency components up to 3*9 = 27Hz in the y direction
    sumY = sum(freqImg(2:27,1));
```

```
    % Standard deviation in the x direction
    if IndX>1
        stdX = std(freqImg(1,2:IndX));
    else
        stdX = 0;
    end
    % Standard deviation in the y direction
    stdY = std(freqImg(2:27,1));
    % Center of gravity in x direction
    if IndX>1
        tx = 1:IndX-1;
        CoGX = freqImg(1,2:IndX)*tx'/sumX;
    else
        CoGX = 0;
    end
    % Center of gravity in y direction
    ty = 1:26;
    CoGY = freqImg(2:27,1)'*ty'/sumY;


    % Set outputs
    outputVector(1,1) = IndX;
    outputVector(1,2) = MagX*100;
    outputVector(1,3) = sumX*100;
    outputVector(1,4) = sumY*100;
    outputVector(1,5) = stdX*1000;
    outputVector(1,6) = stdY*1000;
    outputVector(1,7) = CoGX;
    outputVector(1,8) = CoGY;

end

function fileList = getAllFiles(dirName)

    % Get the data for the current directory
    dirData = dir(dirName);
    % Find the index for directories
    dirIndex = [dirData.isdir];
    % Get a list of the files
    j=1;
    for i = 1:length(dirData(:,1))-1
        if ~dirIndex(i)
            fileList(j,:) = cellstr(dirData(i).name);
            j=j+1;
        end
    end
end

function [memships h] = eval_membership(FV, params, plotON)

    u = zeros(1,length(FV));
    if plotON ~= 0
        figure(1);
    end
    j=1;

    % Loop through clusters
    for c = 1:params.Nrules
        % Loop through features
        for dim = 1:length(FV)
            % Get the beta values
            Bh = params.beta.high(c,dim);
            Bl = params.beta.low(c,dim);
```

```
    midL = params.midL(c,dim);
    midH = params.midH(c,dim);

    % Check if first feature
    if dim == 1
        % Check if indexes match +/- 1
        if abs(FV(dim)-round(params.alpha(c,dim)))<=1
            u(1,dim) = 1;
        else
            u(1,dim) = 0.7;
        end
    else
    % Map using gaussian membership functions e^(-B(x-a)^2),with
    % different beta-values above and below the midpoint.midL and midH
    % shifts the midpoint used so that a flat region on the top
    % is generated on the interval [a-midL,a+midH]
        if FV(dim)>params.alpha(c,dim)+midH
           u(1,dim) = exp(-Bh*(FV(dim)-(params.alpha(c,dim)+midH))^2);
        elseif FV(dim)<params.alpha(c,dim)-midL
           u(1,dim) = exp(-Bl*(FV(dim)-(params.alpha(c,dim)-midL))^2);
        else
            u(1,dim) = 1;
        end

        %% Plot result
        if plotON ~= 0
            % Get x-intervall
            y = 0.005;
            Xl = -sqrt(log(y)/-Bl)+(params.alpha(c,dim)-midL);
            Xh = sqrt(log(y)/-Bh)+(params.alpha(c,dim)+midH);

            x1 = Xl:0.1:Xh;
            % Set gaussian value vector to the same length as x1...
            gauss = x1;
            %...and compute the function e^(-B(x-a)^2), with different
            %beta-values above and below the midpoint. midL and midH
            %shifts the midpoint used so that a flat region on the top
            %is generated on the interval [a-midL,a+midH]
            for i=1:length(x1)
             if x1(i)<params.alpha(c,dim)-midL
             gauss(i) = exp(-Bl*(x1(i)-(params.alpha(c,dim)-midL))^2);

             elseif x1(i)>params.alpha(c,dim)+midH
             gauss(i) = exp(-Bh*(x1(i)-(params.alpha(c,dim)+midH))^2);

             else
             gauss(i) = 1;
             end
            end
            % Plot and set plot parameters
            h(j) = subplot(params.Nrules, 7, j);
            plot(x1,gauss);
            xlim([Xl Xh]);
            line([FV(1,dim) FV(1,dim)],[0 u(1,dim)],'Color','red');
            line([Xl FV(1,dim)],[u(1,dim) u(1,dim)],'Color','red');

           text(Xl+0.1,u(1,dim)+0.03,num2str(round(u(1,dim)*100)/100),
           'Color','red','Fontsize',9);
            j=j+1;
        end
    end
end
% Check status of feature 1
if plotON ~= 0
```

```
            if u(1,1) == 1
                title(h(c*7-2),'X-frequency peak index OK','Fontsize',14);
            else
                title(h(c*7-2),'X-frequency peak index NOT OK',
                'Color','red','Fontsize',14);
            end
        else
            h = -1;
        end
        % AND using product, AND using minimum, AND the two values using
        % product
        memships(c) = prod(u)*min(u);
    end
end
```

APPENDIX D

Figures 1D – 20D contains images of defects along with the result of the Matlab algorithm.

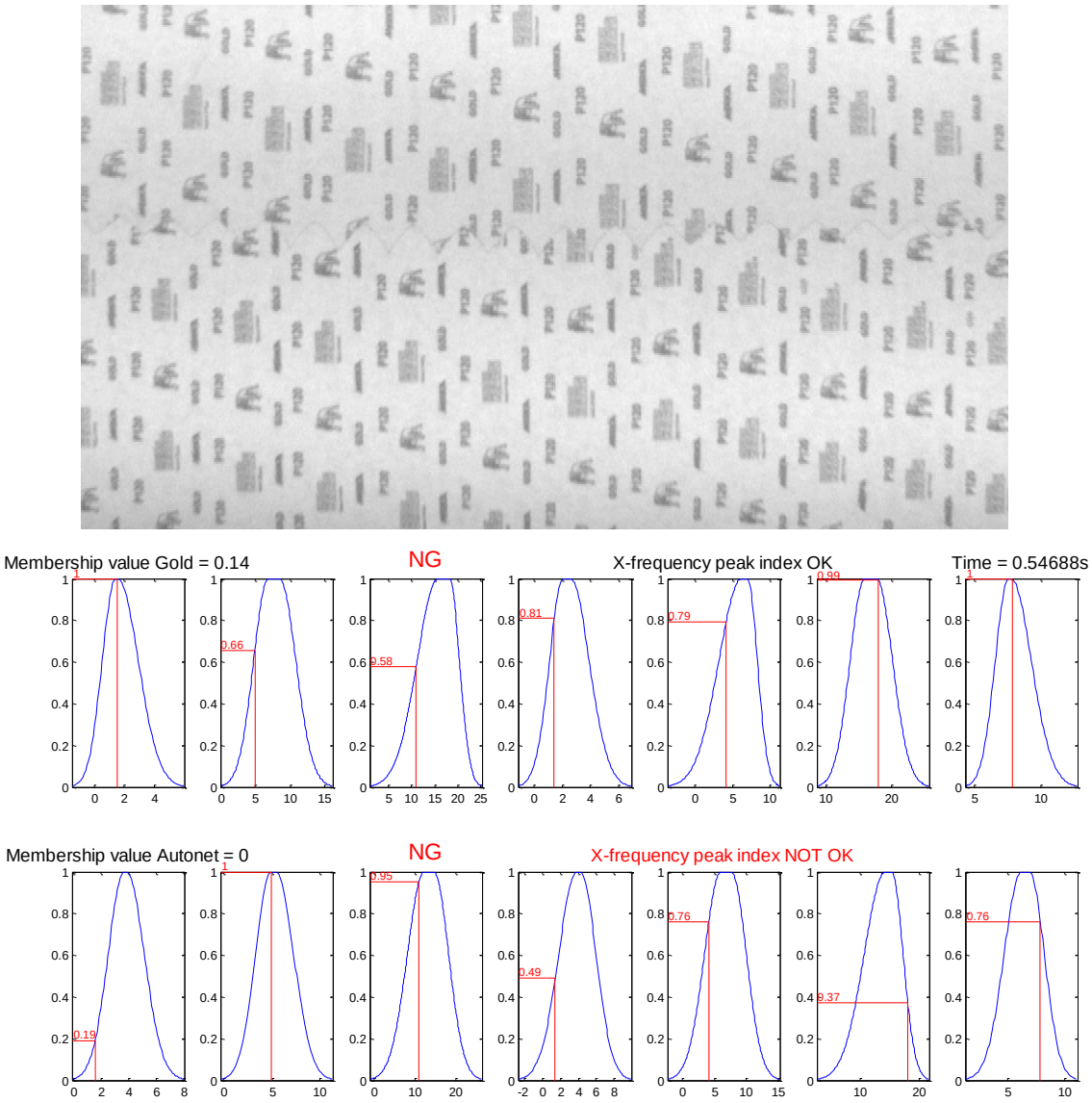
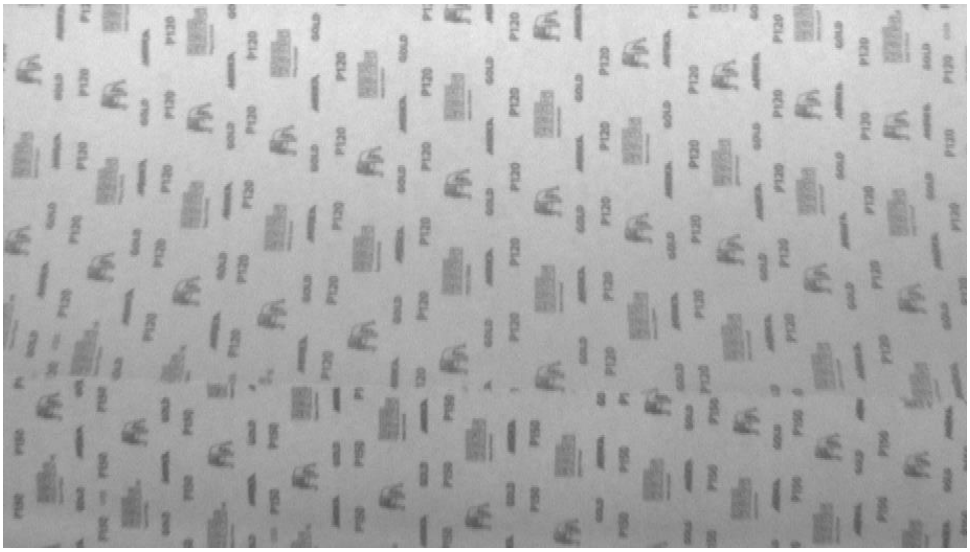


Figure 1D. This joint was captured the same time as the five problematic faultless images of Gold, which already has low memberships. However, it has a membership of 0.14 which is lower than all faultless images of Gold.



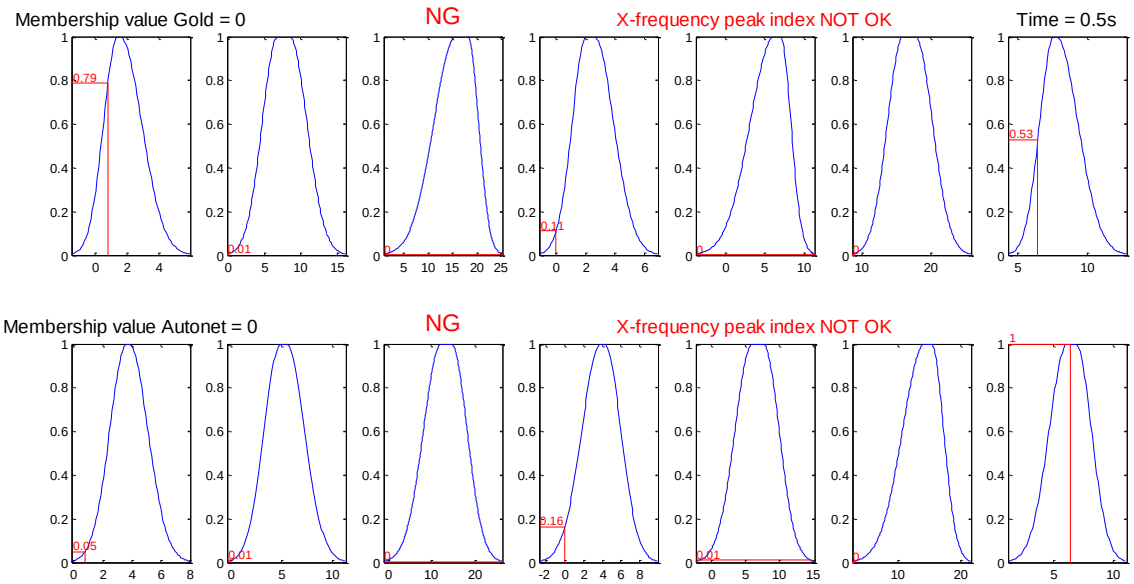


Figure 2D. The edge of this joint is loose and bends away from the surface. Both shadows and change in pattern occur and the membership calculated is zero.

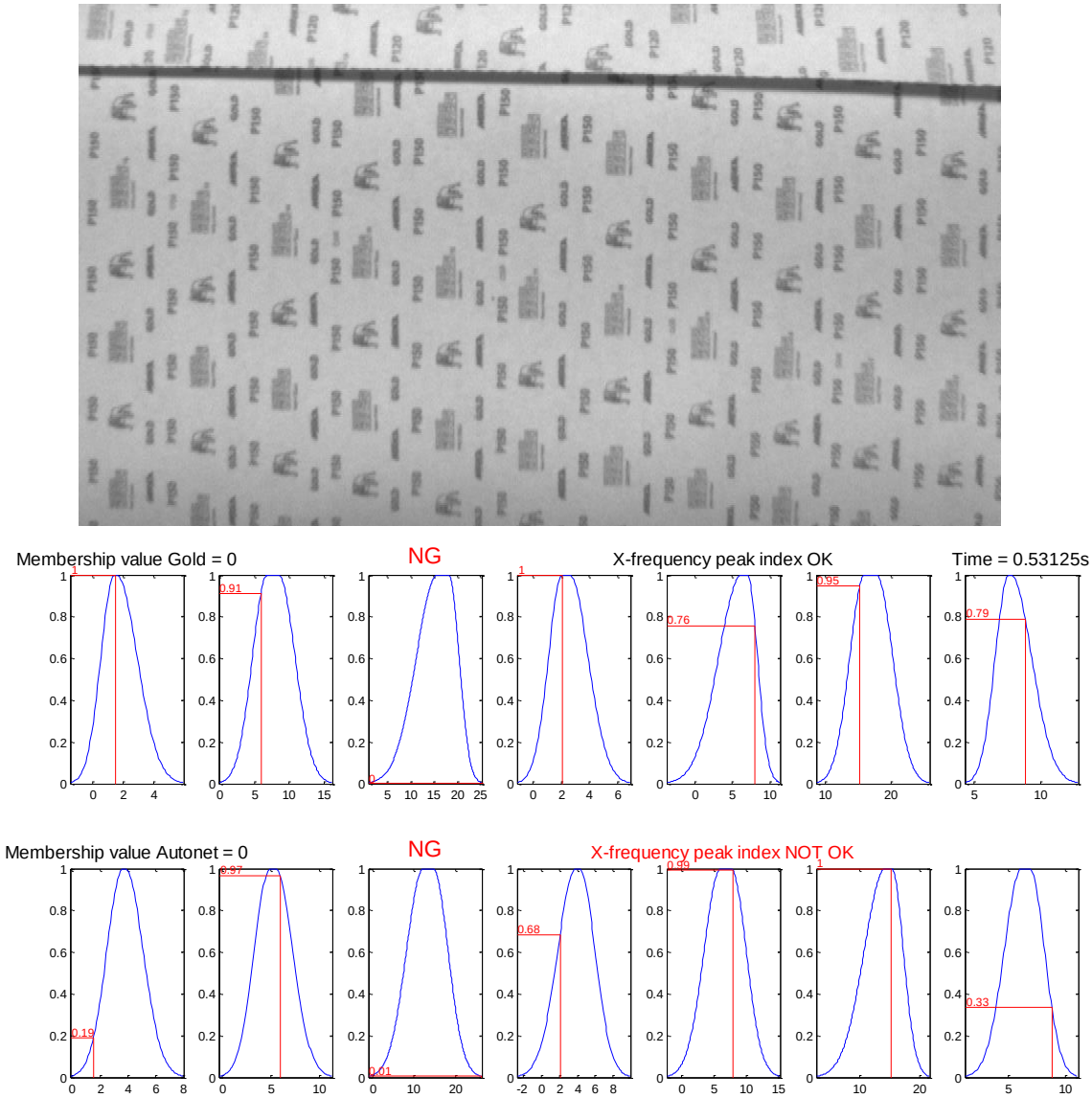


Figure 3D. The same defect as in Figure 56 has moved upwards and now casts a shadow visible as a dark band. The membership is still zero, but the feature values have changed significantly.

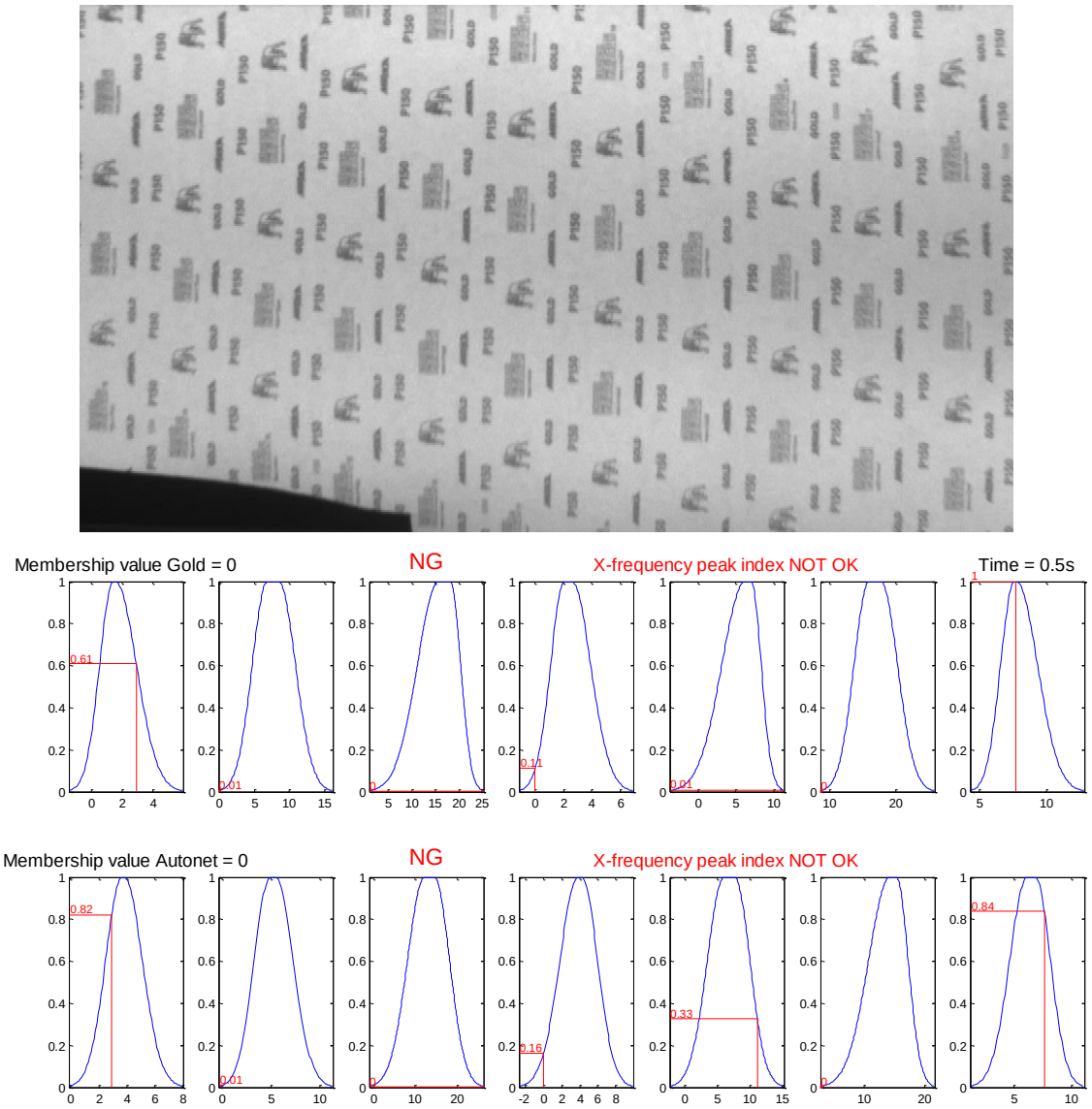


Figure 4D. A large hole is captured as it enters the imaging area. Membership is zero.

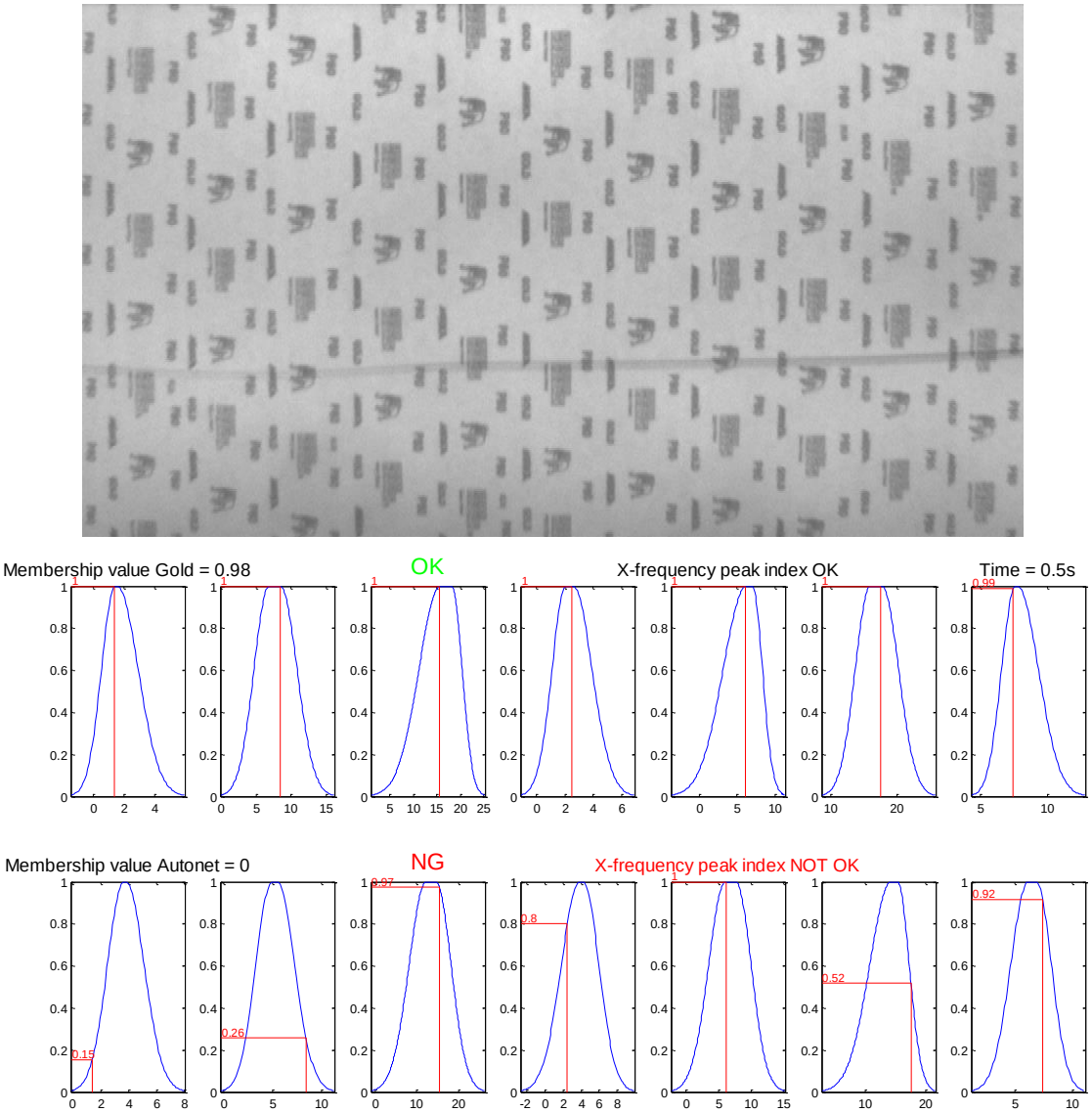


Figure 5D. This line alone does not change the pattern significantly and is therefore not detected at all.

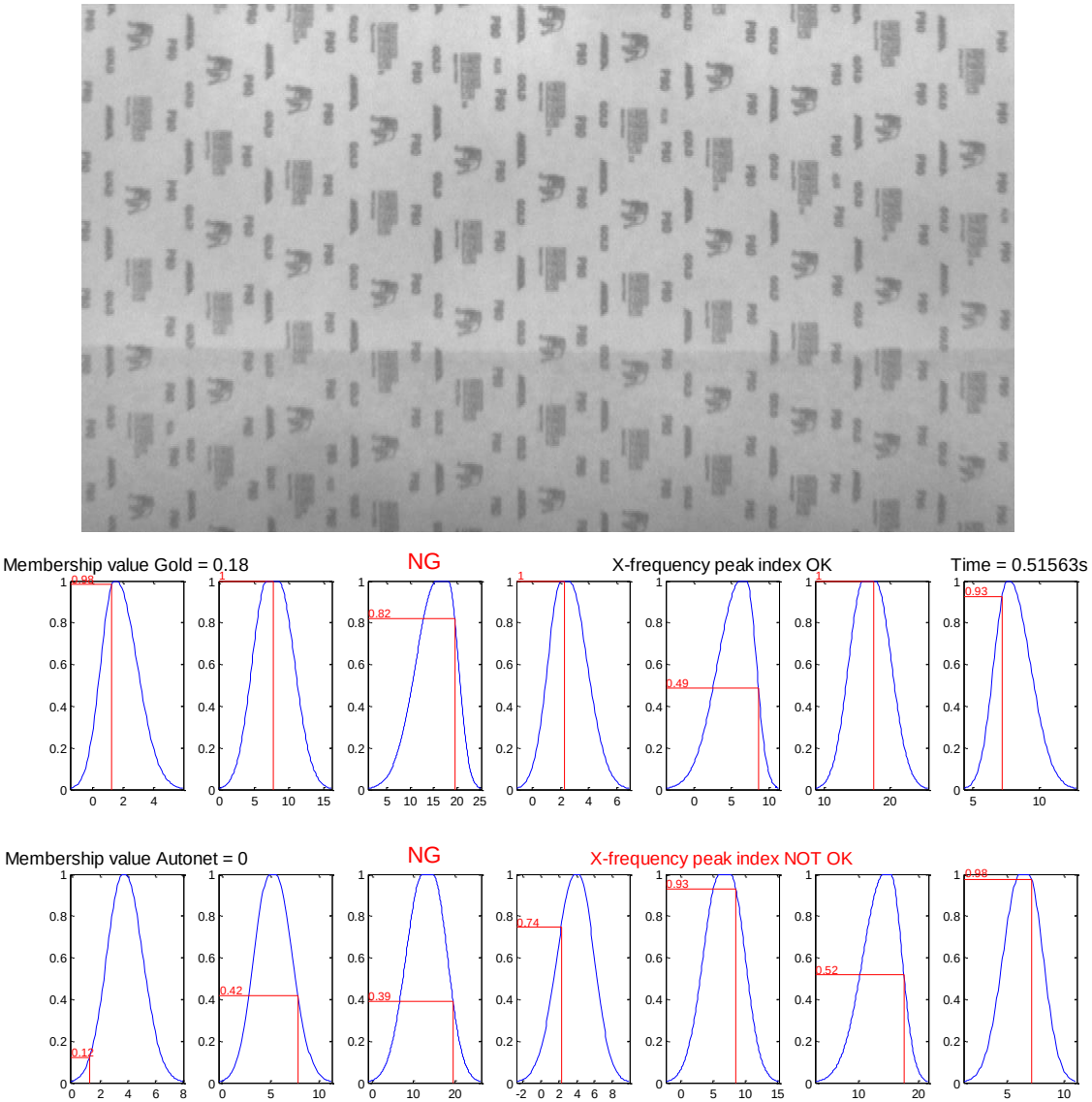


Figure 6D. This joint happens to preserve the same pattern between the both sides, but the surface intensity differs. This causes feature 6, standard deviation of frequency power in y-direction, to increase and generate a low membership value. Total membership is at 0.18.

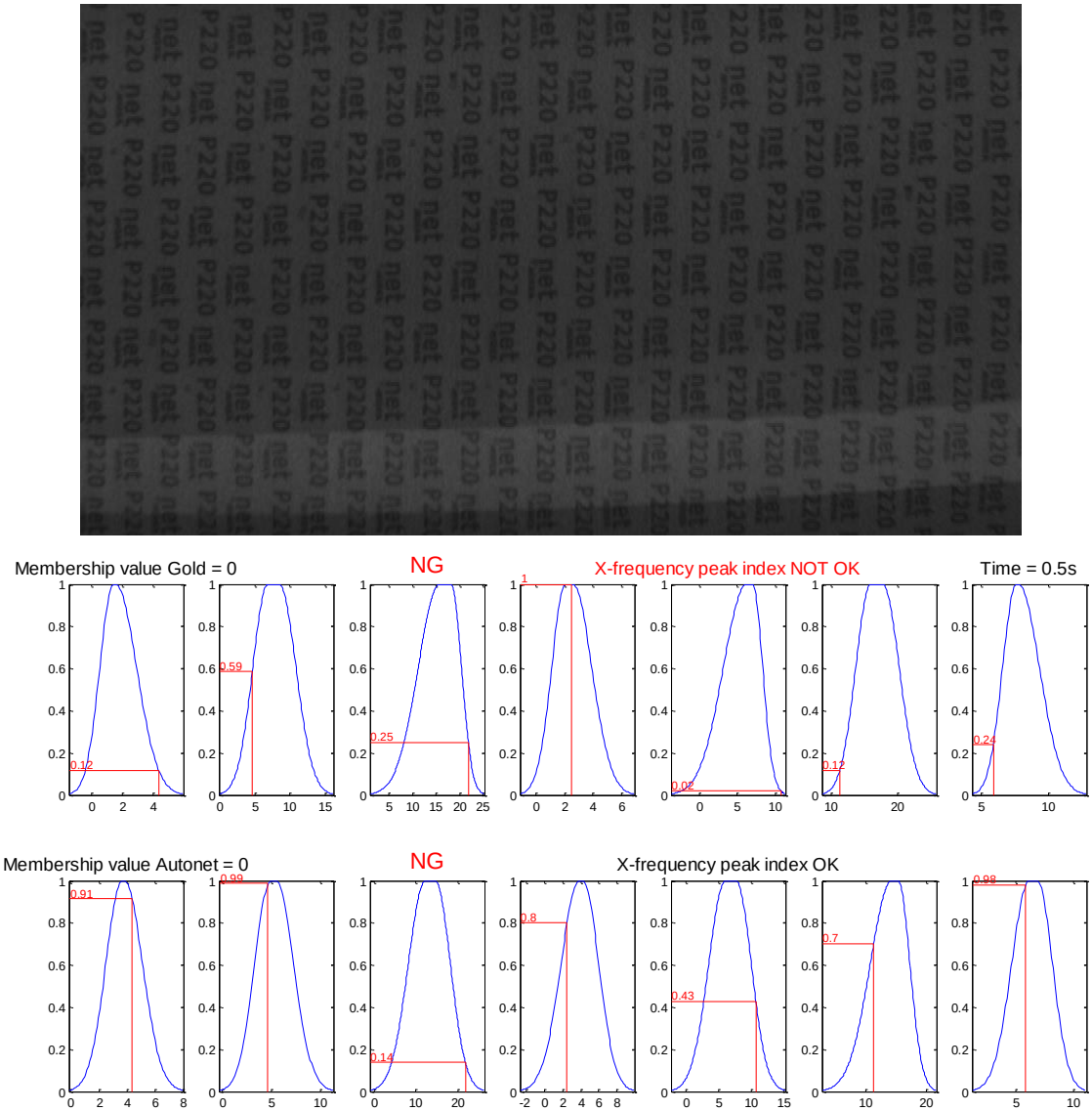


Figure 7D. This defect is reliably detected with a membership value of zero in every image. So are also all other bright joints.

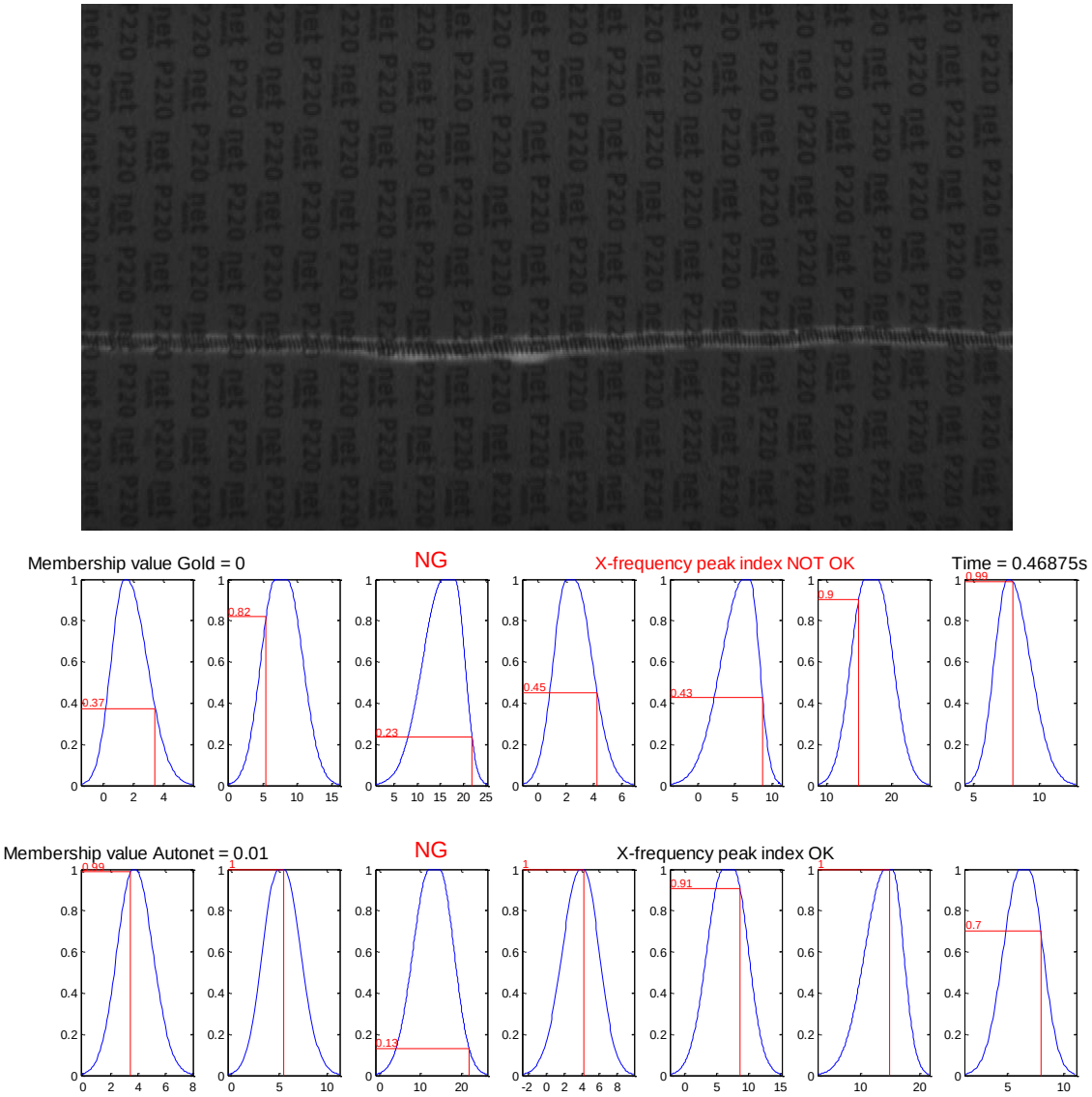


Figure 8D. This defect is also reliably detected in every image with memberships near zero.

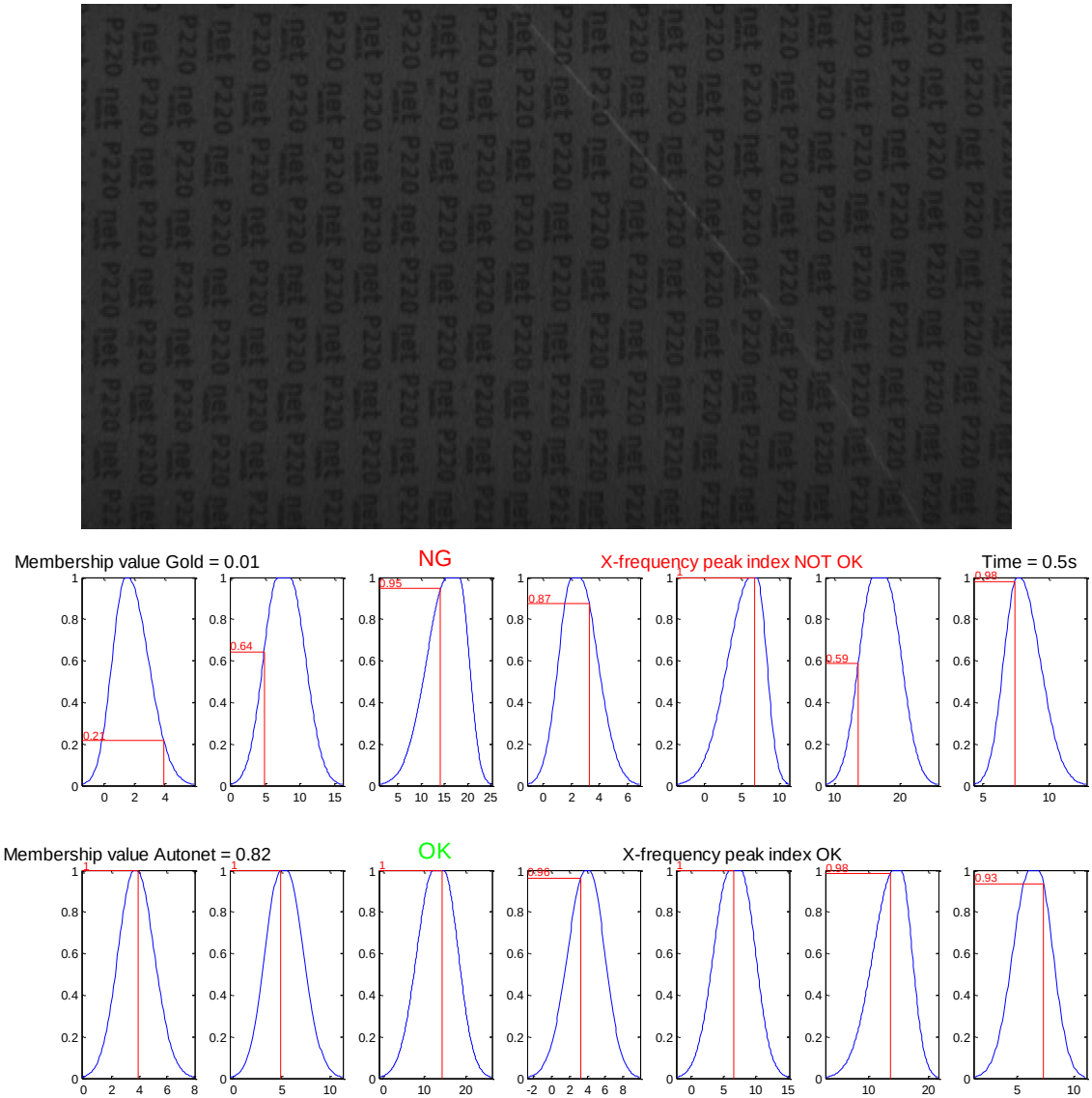


Figure 9D. This single bright scratch does not alter the pattern enough and is therefore not detected.

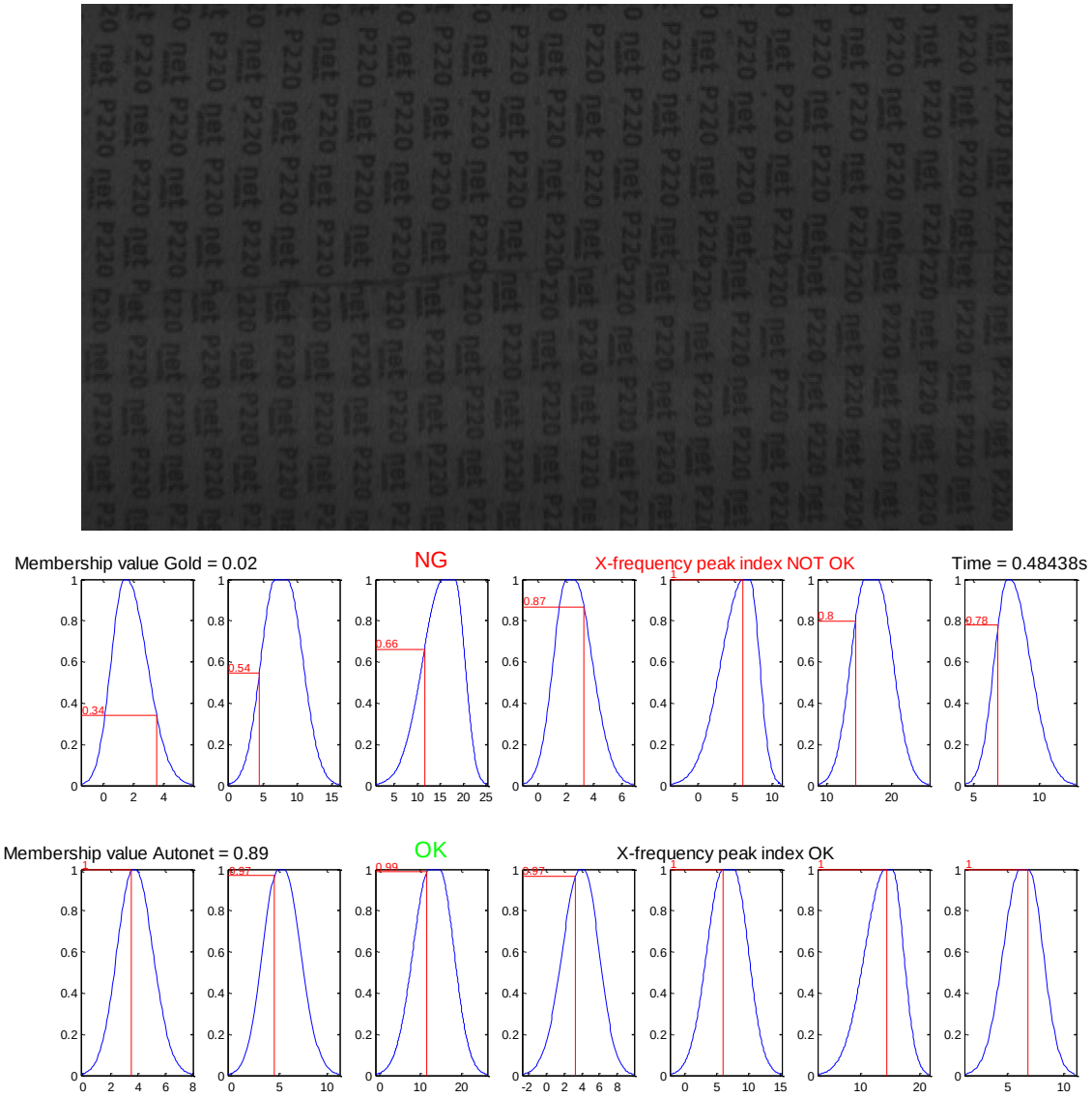


Figure 10D. This fold was not detected in this particular image, but in the next one, shown in Figure 11D, the membership value drops significantly. The drop is probably due to the brighter region following the fold. This fold happens to not change the relative positions of the printings much. Compare this fold to the one in Figure 13D.

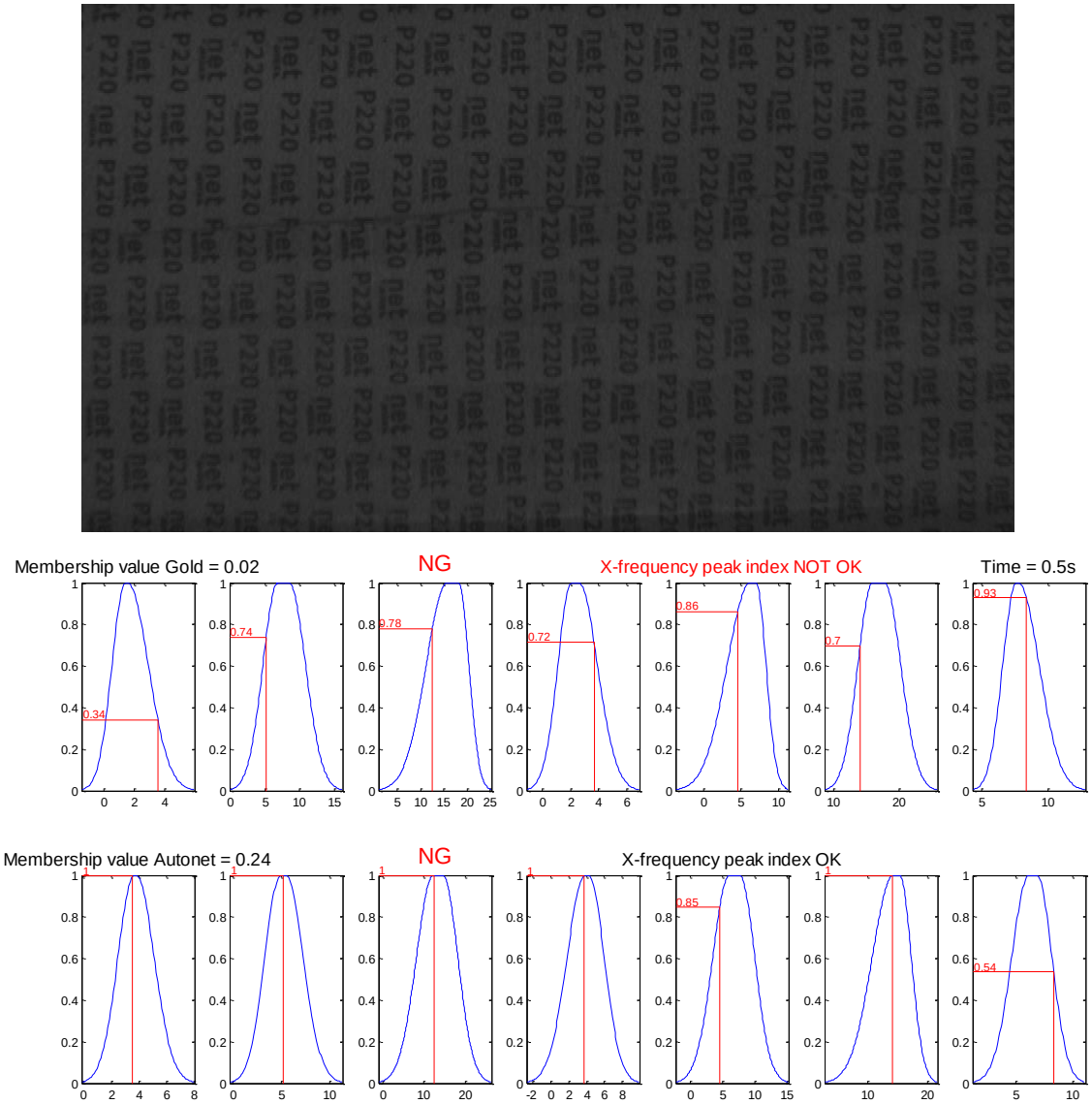


Figure 11D. The same defect as in Figure 10D is now assigned a lower membership value of 0.24. The brighter area following the fold is probably the cause. This area is visible in the next figure.

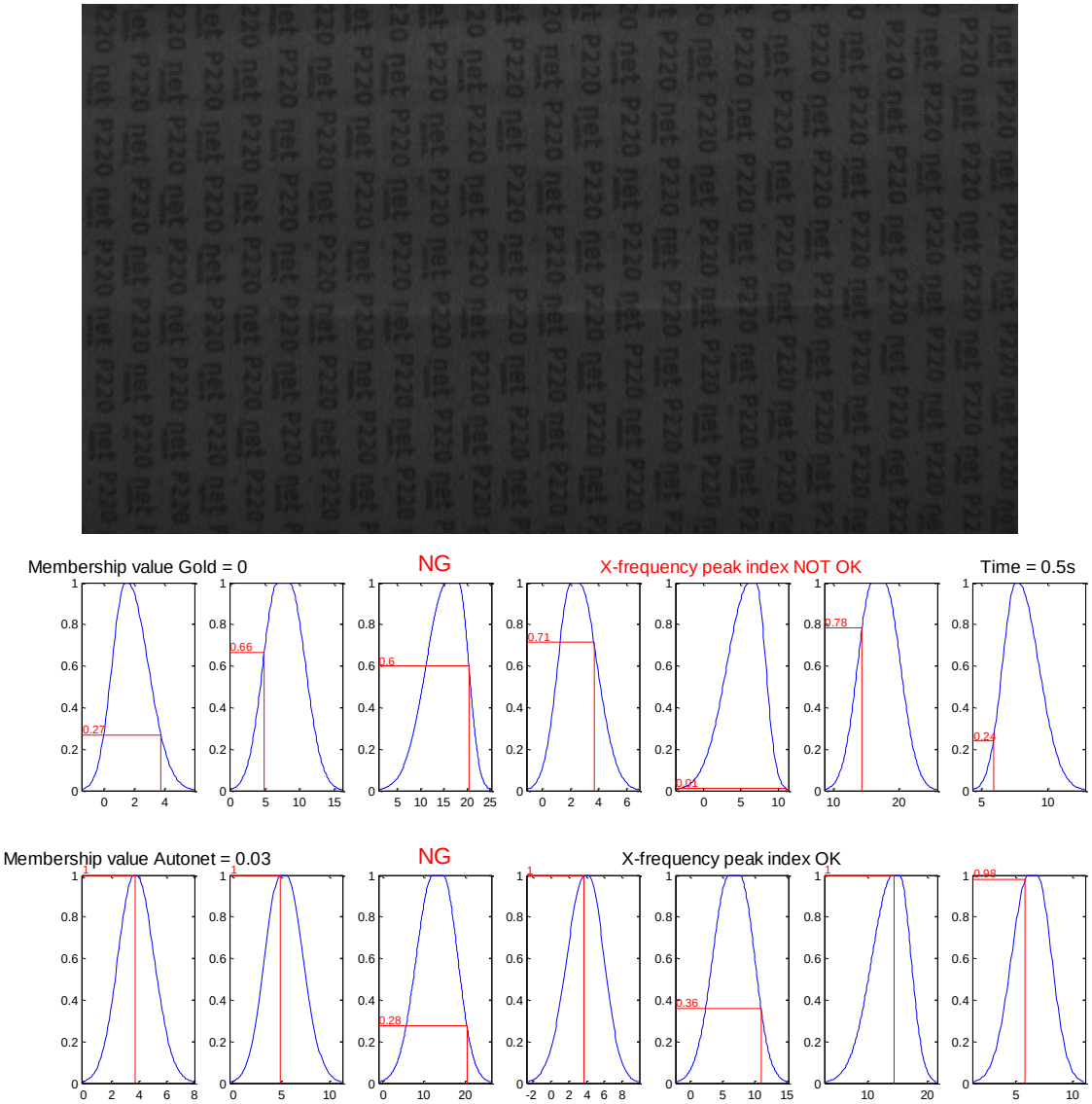
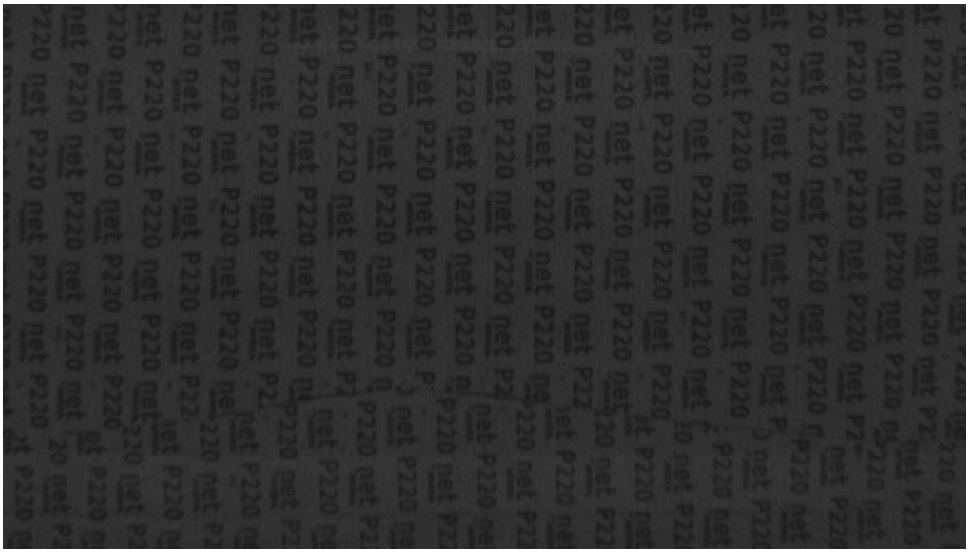


Figure 12D. The brighter area following the fold shown in Figures 10D and 11D causes the membership value to drop.



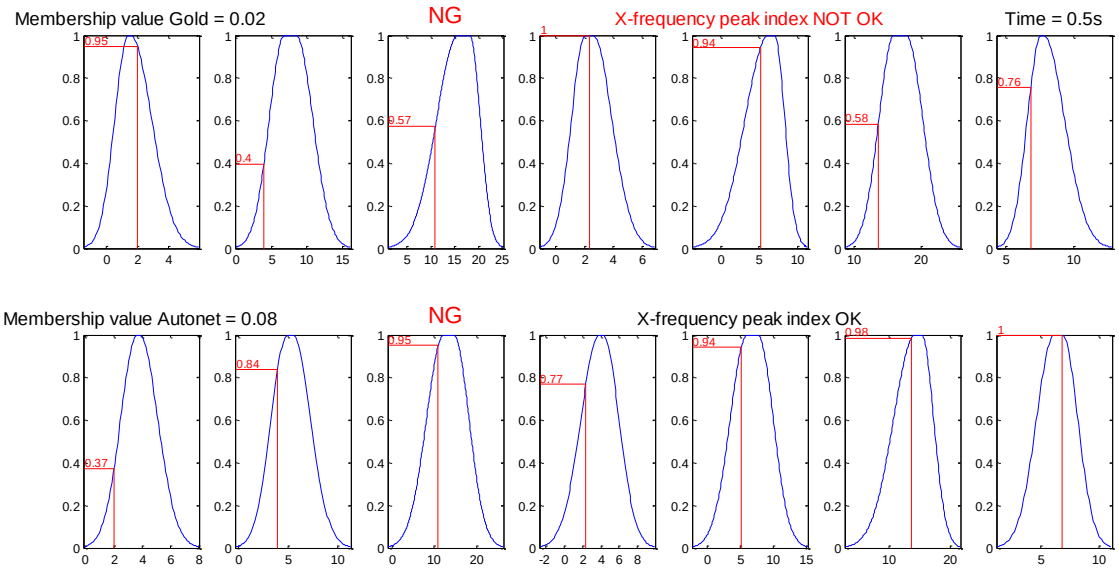
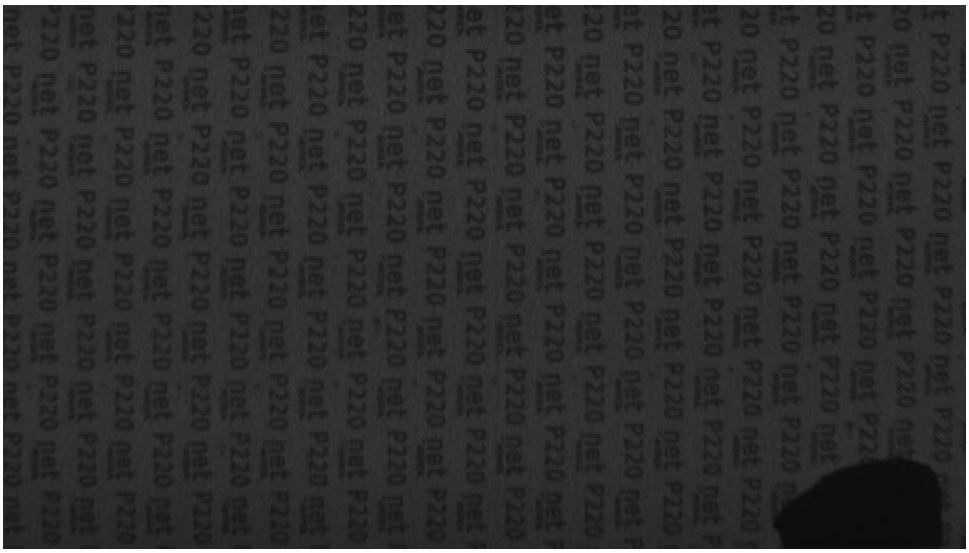


Figure 13D. This fold shifts the printings in both x- and y-directions and has therefore a greater impact on the frequency spectrum than the fold in Figures 10D and 11D. Membership drops to 0.08.



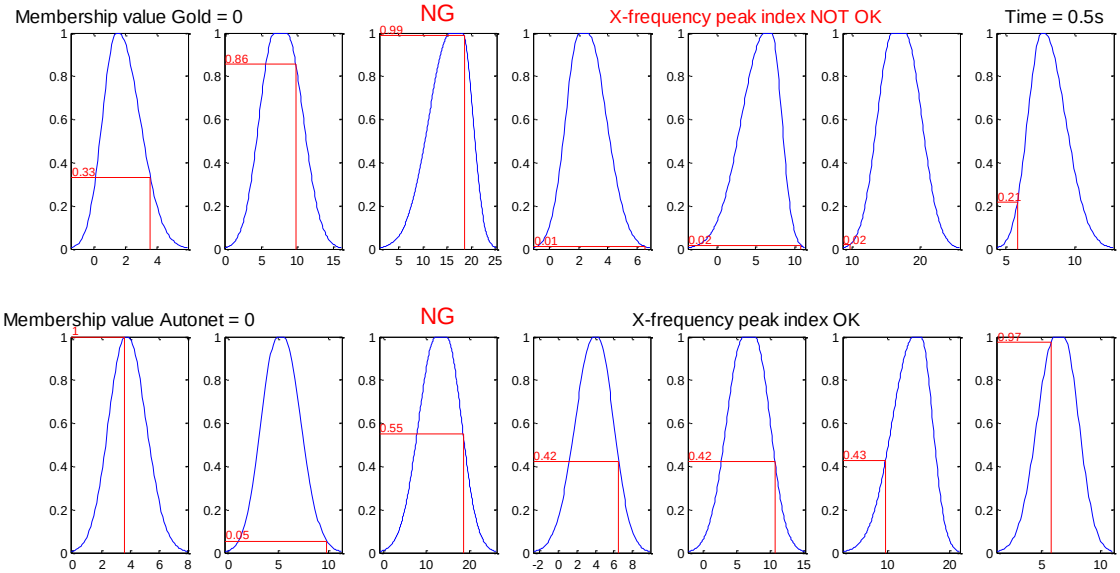


Figure 14D. A hole in Autonet material causes the memberhip value to drop to zero.

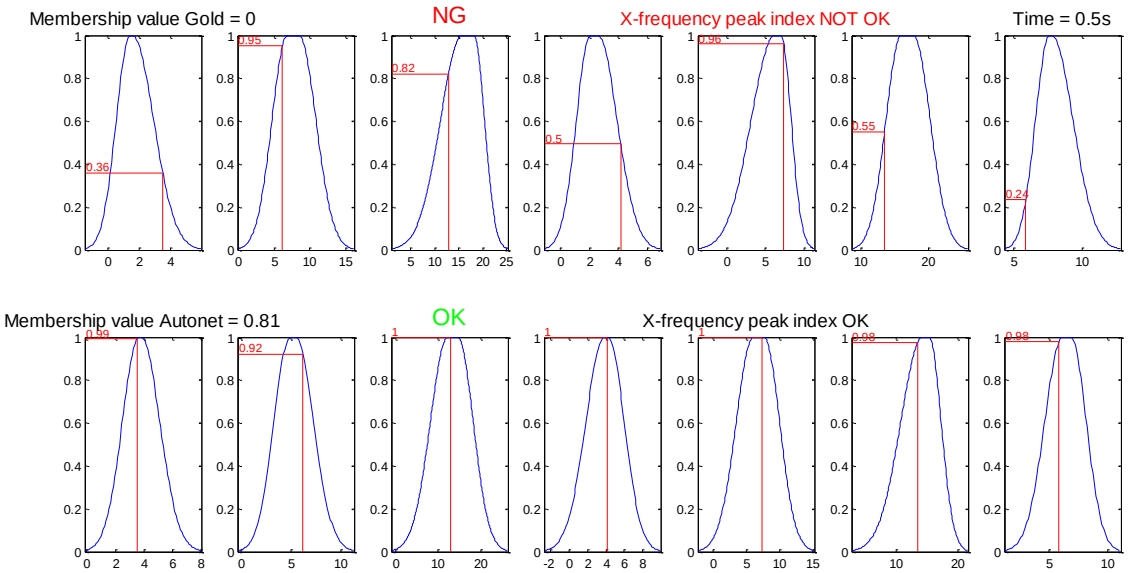


Figure 15D. A small amount of scratches in the material does not alter the membership value significantly. However, the amount of scratches and their relative positions can cause lower memberships in some images. Compare to the two following figures.

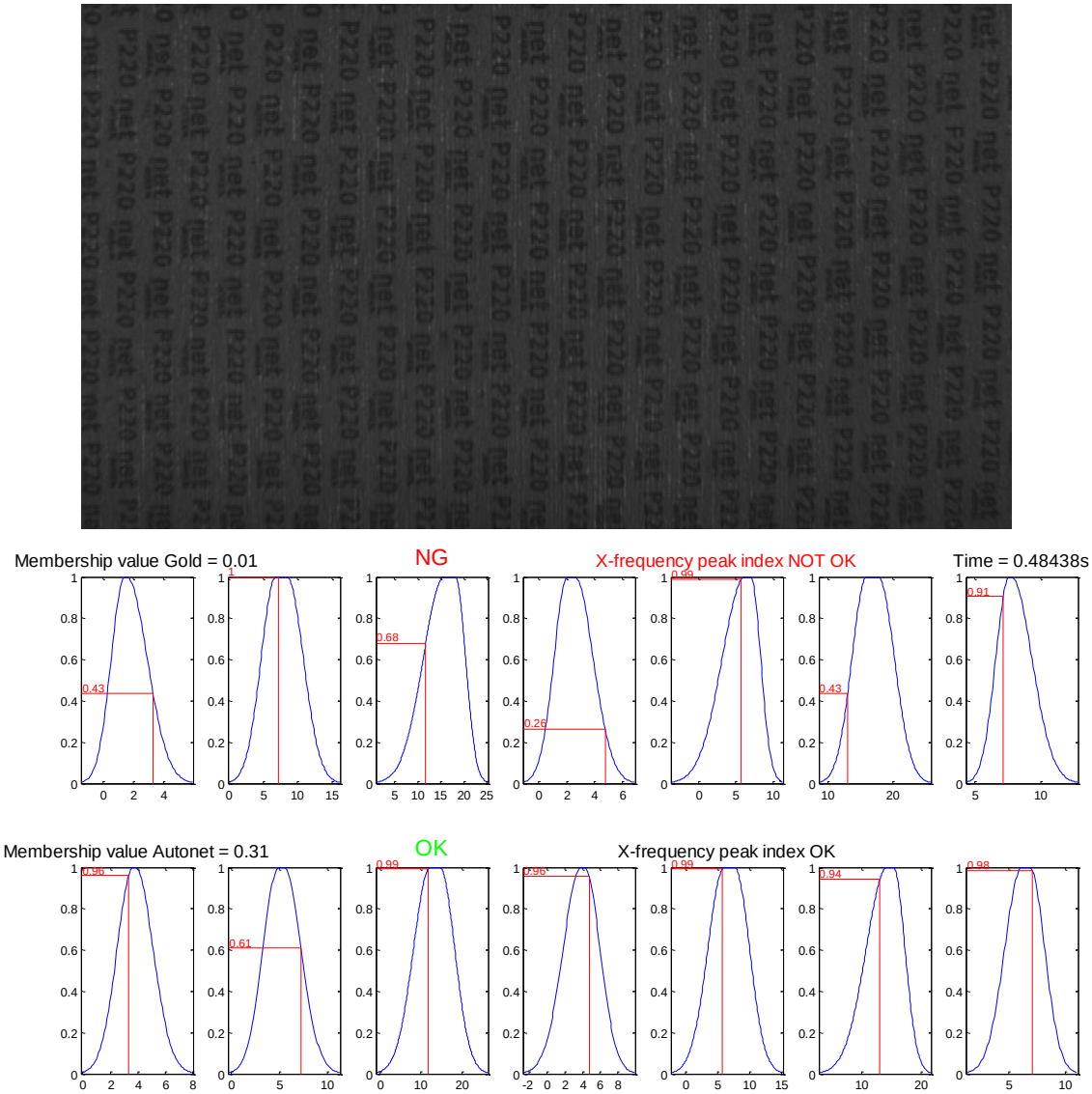


Figure 16D. In this image the scratches causes feature 3, sum of frequency power in x-direction to increase. The membership value drops to 0.31. There are quite many scratches in this image compared to the next one.

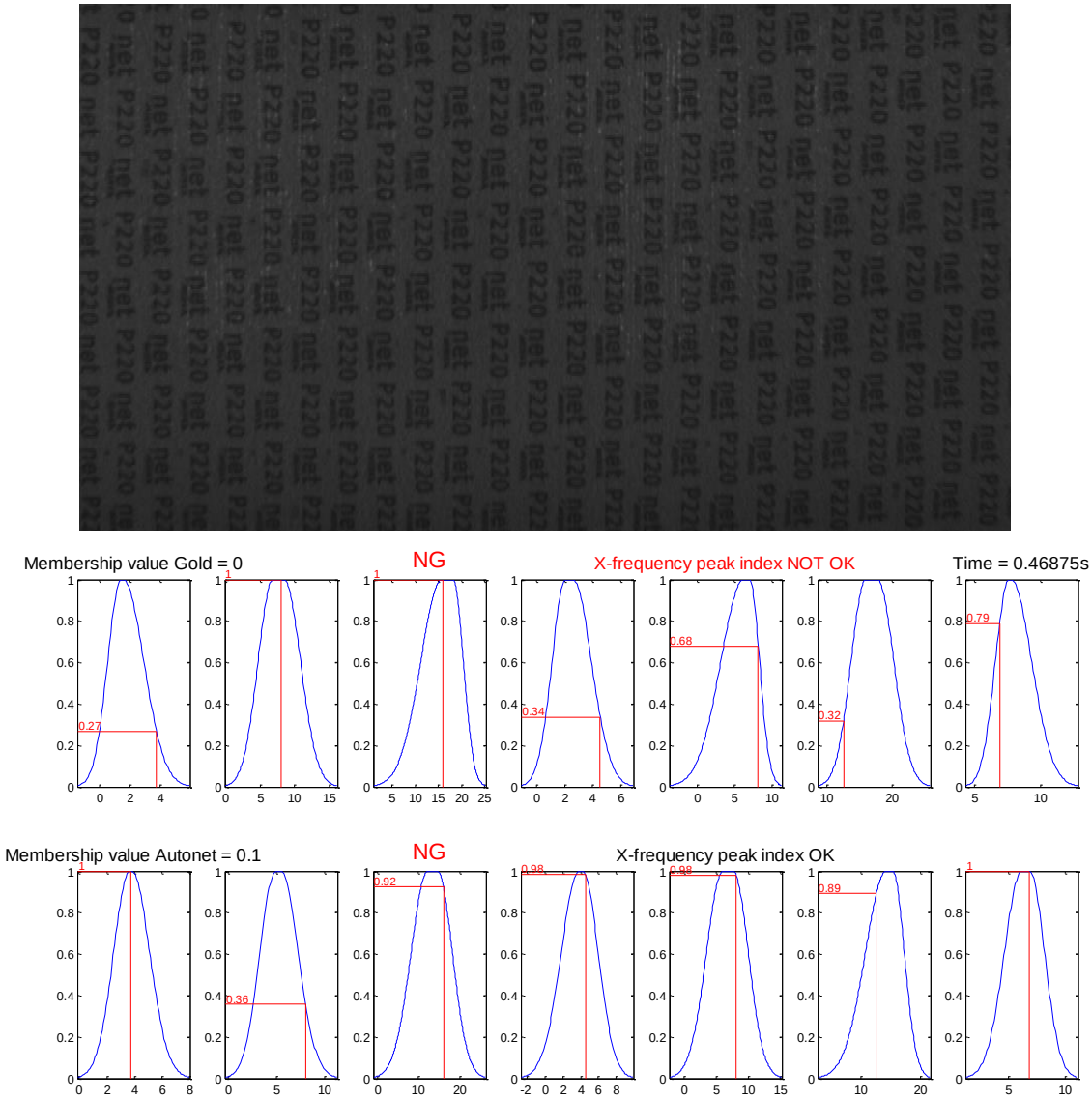


Figure 17D. Scratches are fewer in this image than in the previous one, but the membership is still lower. This indicates that the algorithm is more sensitive to the relative positions of the scratches than to the number of them.

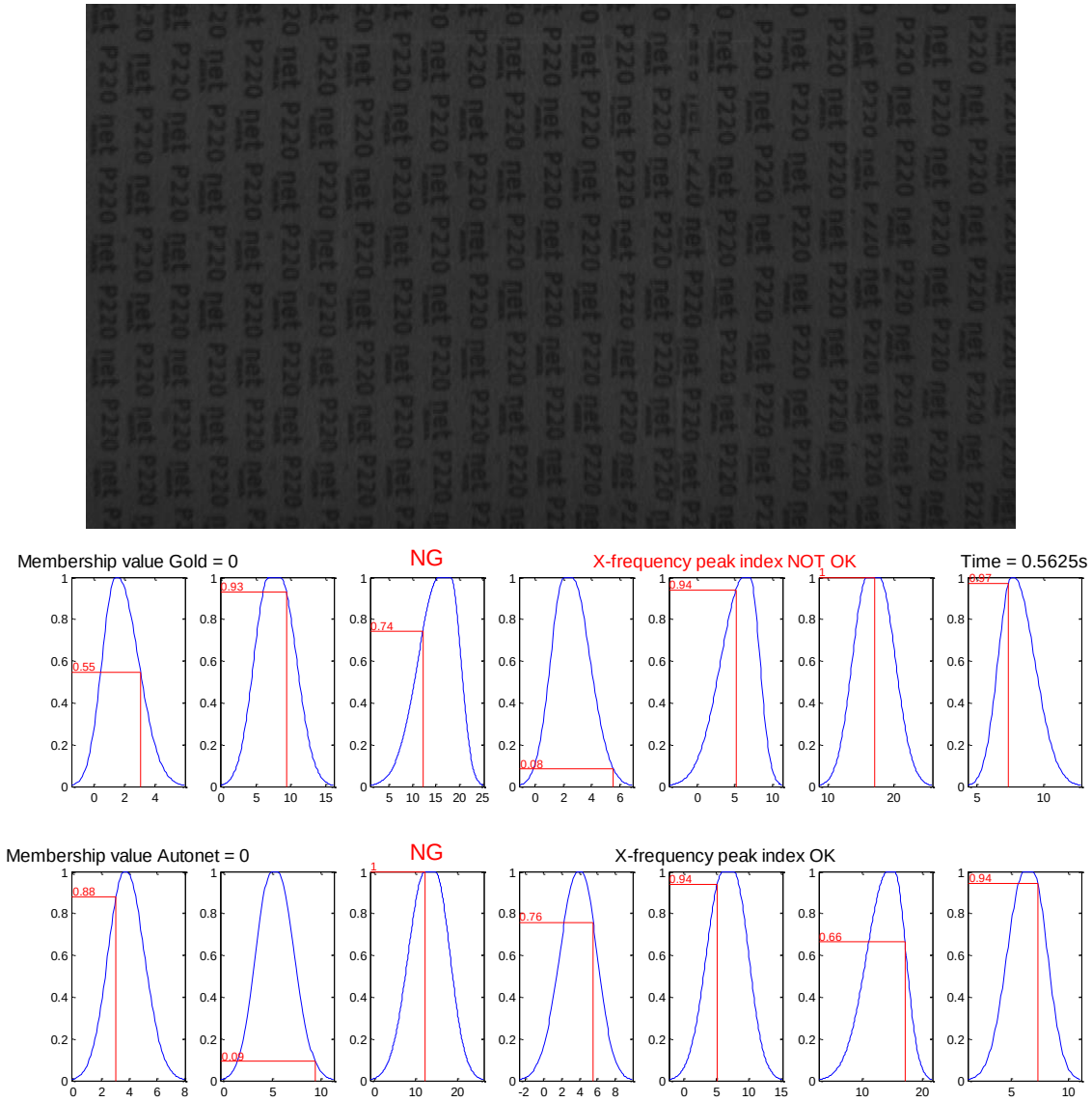


Figure 18D. Folds that change the relative positions of printings cause the algorithm to react strongly. About 20 images of similar folds to these have been captured and the majority of them have a membership of zero. All have memberships below 0.13.

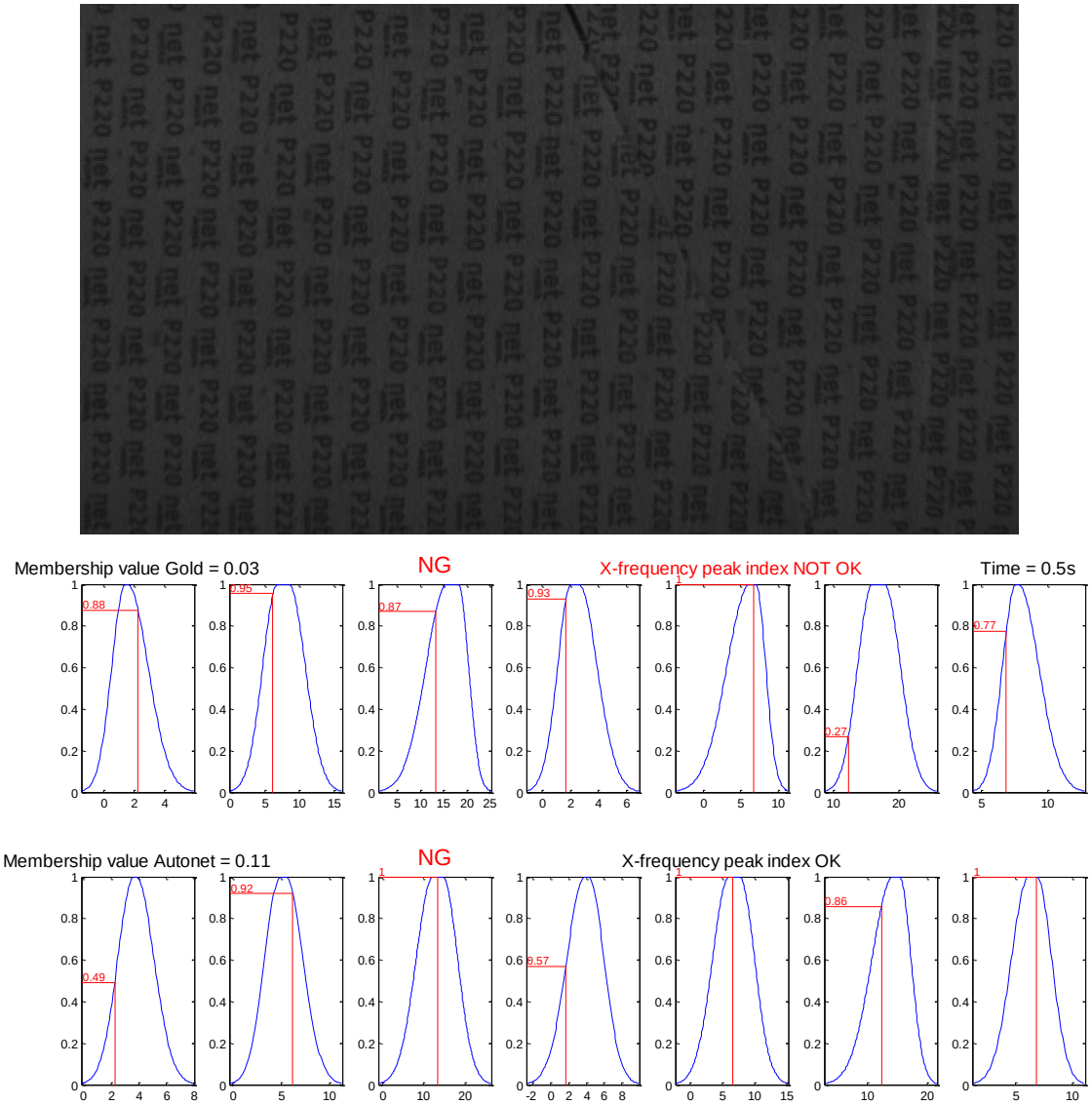


Figure 19D. Another fold with a membership of 0.11.

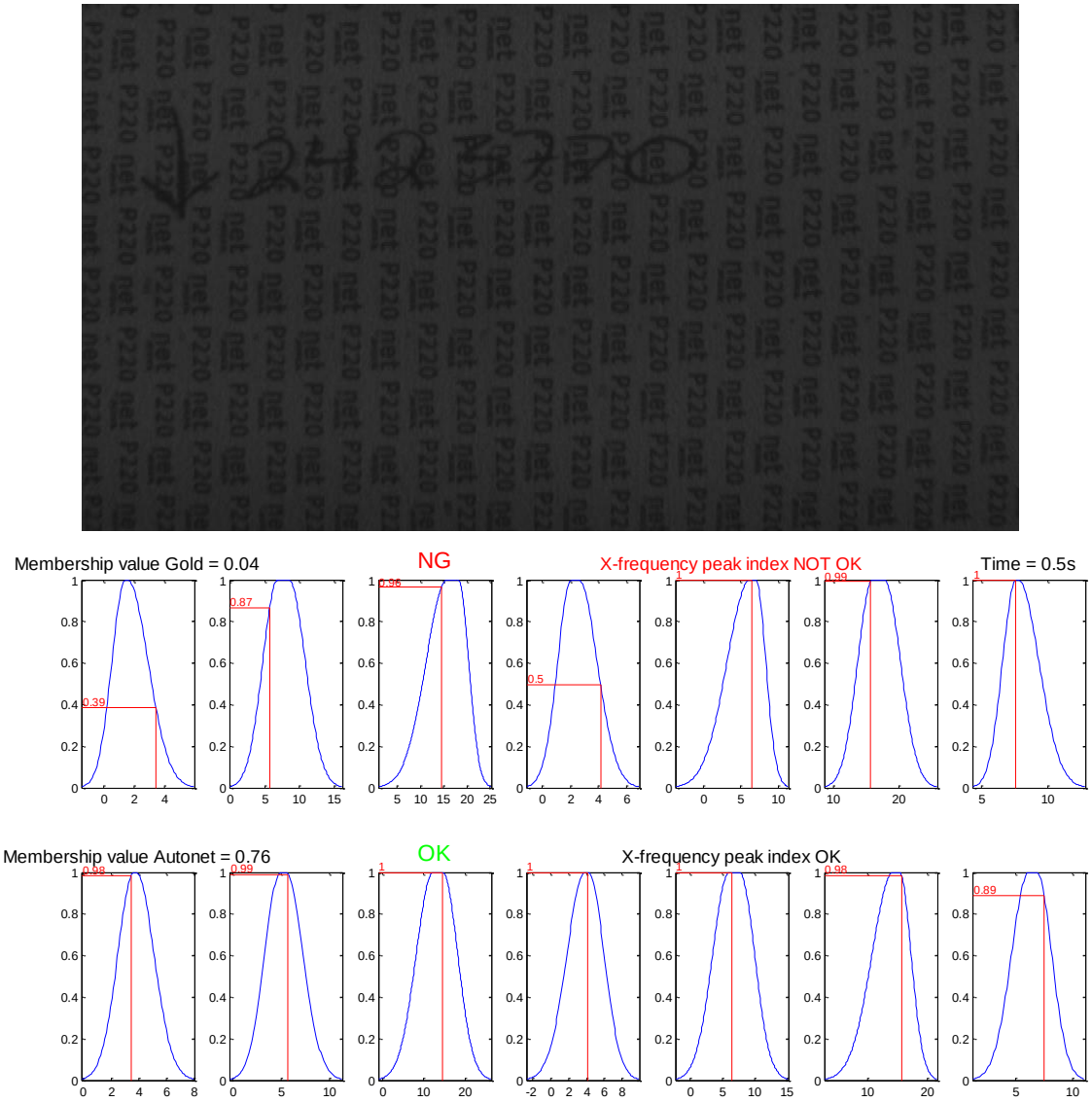


Figure 20D. In a single image of this defect, the membership value was calculated to 0.09. In other images of the same defect however, the values are much higher and the detection is therefore not reliable.