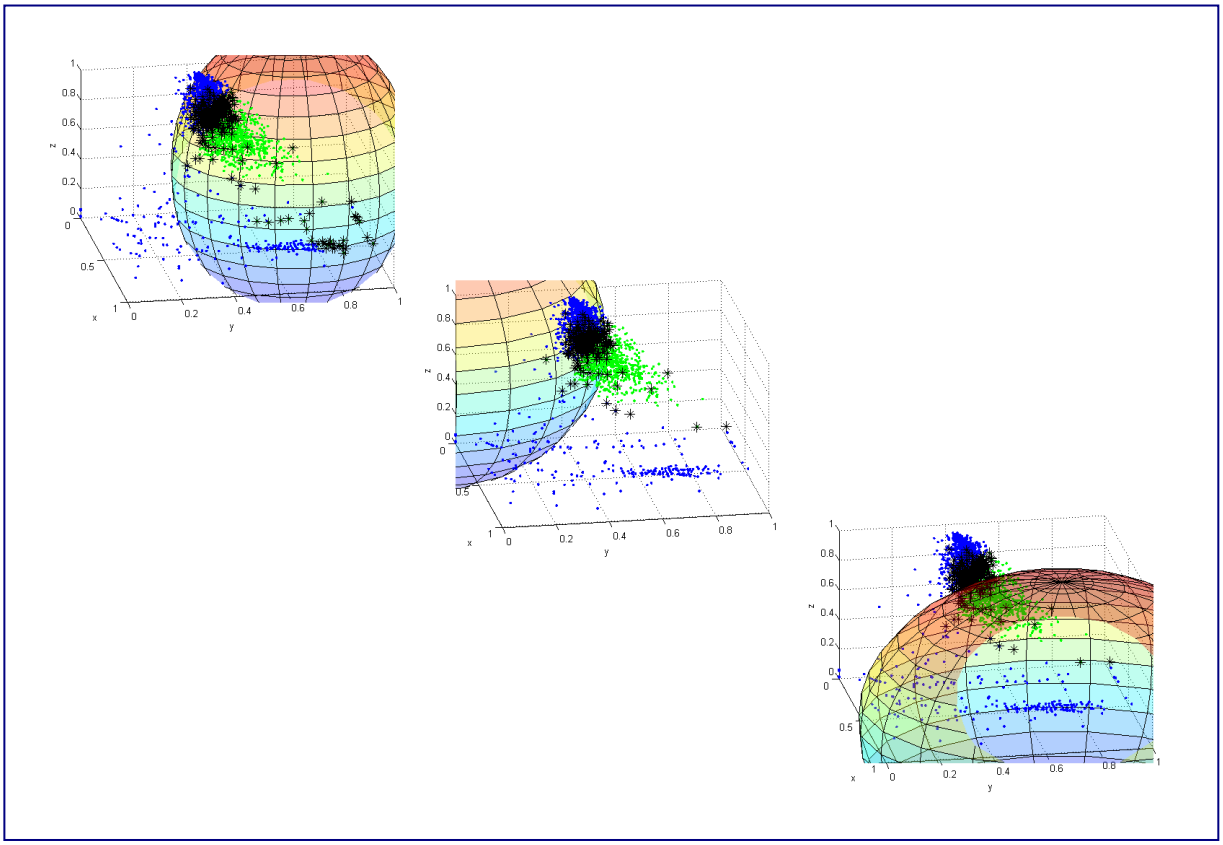
CHALMERS



Machine Vision for Wood Defect Detection and Classification

Master of Science Thesis

SELMAN JABO

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2011

Report No. Ex006/2011

Report No. Ex006/2011

Machine Vision for Wood Defect Detection and Classification

Master of Science Thesis

SELMAN JABO

Supervisor Professor Irene Y.H. Gu

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2011

Machine Vision for Wood Defect Detection and Classification

Master of Science Thesis
SELMAN JABO

Report No. Ex006/2011
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

Department of Signals and Systems
Göteborg, Sweden 2011

Machine Vision for Wood Defect Detection and Classification

Master of Science Thesis
SELMAN JABO
Department of Signals and Systems
Chalmers University of Technology

ABSTRACT

Sawmills deliver wood after standardized rules for quality assessment. The wood quality sorting is based on machine vision. Image analysis and classification methods are proposed which cover the sorting of defects such as “wood decay”, “blue stain” and “shake”. A supervised classifier is first trained with Adaboost and then used to extract colors of stain type defects and methods such as integral image, Dynamic programming and Hough transform are combined to extract features from image objects. The proposed methods are chosen to maximize the classification rate (with low false alarms), meanwhile to minimize the computation time which is an important factor in industry applications. Experiments have been conducted on timber images, and results demonstrated the effectiveness of the proposed approach.

Key words: Machine Vision, Artificial Intelligence, Image Processing, Image Analysis, Classification., Adaboost, Wood defects

En Maskin Baserad Defekt Detektering och Klassificering av trävirke.

Masterexamensarbete vid Signaler och system
SELMAN JABO
Institutionen för Signaler och System
Chalmers Tekniska Högskola

SAMMANFATTNING

Sågverk följer standariserade sorteringsregler när de kvalitetsbedömer sitt trävirke innan de levererar till sina kunder. I detta examensarbete ges förslag om hur man kan använda sig av bildanalys i en maskinbaserad kvalitetsbedömning av trävirke som kan känna igen Röta, Blånad och Sprickor. Ett klassificerings program tränas med Adaboost för att känna igen färger som tillhör stora ytdefekter som Röta och Blånad. Med hjälp av Integral image, Dynamic program och Hough transform kan man utvinna de karaktersitiska dragen från ett bildobjekt. Metoderna som användes är valda för att minimera berkäningstiden och det gynnar hastigheten som är en viktig faktor inom industrin.

Nyckelord: Bildanalys, Klassifisering, Artificiell intelligens, Maskinbaserad Vision, Adaboost, Trä defekter.

Acknowledgment

These studies led me into the thoughts of how we humans are able to understand what we see and translate that knowledge into a digital language. With the help and support from Erik Eveborn Director of Rosén & Co Maskin AB (Roséns), I had the opportunity to go for a field study and meet up with people that are experienced with timber and its defects. I would like to thank Bertil Stenman, Engineer from SP Wood Technology, and Henrik Eriksson, Engineer from Roséns, for taking the time to explain some valuable hints to recognize defects. I would like to thank my supervisor Irene Y.H. Gu, Professor at the Department of Signals and Systems, for giving me the opportunity and helping me getting started with this master thesis. I would also like to thank Erik Eveborn for providing me with images on Timber from Roséns and financing this work.

Göteborg Januari 2011

Contents

1 About This Thesis

1.1 Background

The interests in robotic automation in industries are growing and new methods and robots are developed continuously. In the tree industry, the saw mills speed of production and time of delivery can be related to its revenues. The automation in sawmills production-lines has been available in many decades but in some areas it's difficult to have automation, especially by utilizing a robot with the ability of visually inspecting the product and rate it with right pace of work. Automated quality controls are available in industry e.g. control of egg quality by visually analyzing the, size, color or search for defects and also for visually analyzing electronic circuits, if there is any defect e.g. short circuit, a bad welding, etc. In many sawmills, the board quality is still controlled by a human sorter because the defects can vary in many ways, color, area, shape, amount and the type of defects can be in various forms. It is important for sawmills to deliver right quality of boards to their clients so the sorting is depending on the worker. The work can be very tedious and it can be expected that a worker makes some mistakes. The small amount of sorting errors can lead to losses in income for sawmills.

The earliest form of a computer used in the industry where the programmable Jacquard loom that was invented by Joseph Marie Jacquard in 1801. He used punched cards for the loom to create textiles with complex patterns. Since then, the machines have evolved dramatically without slowing down in their progress. The speed and efficiency of a machine are far superior to humans in many fields we might even have to consider if the machines will be able to outsmart humans in the future. The goal of artificial intelligence (AI) is to reach the human level of intelligence and beyond so that machines can understand and solve problems [McCarthy 2007]. The use of AI has already been assisting us humans in economy and military, today many stock trades are initiated by computer programs that analyze the stock market and understands the changes that occur in the market in a similar way as human day trader does, within milliseconds a program can make decisions whether to buy or sell shares [Mallick et al., 2008].

The purpose of this thesis is to automatically detect and classify wood defects from images captured from the visible spectrum. This is done by finding the right methods to use for extracting features and patterns from images and then with a decision making program one can analyze the information so that a decision can be made.

Just as a person would visually qualify a timber by recognizing a wood defect, the program is first trained to recognize the colors and the patterns of wood defects and then based on the information it has acquired during the training mode, the program will make a decision.

1.2 Purpose

The purpose of this master thesis is to investigate and develop methods that can be utilized in a quality control program. The program scans a sorting line for boards and visually inspects each timber with a set of CCD cameras that operates in the visible spectrum. The program will search for wood defects such as blue stain, shakes and wood decay. A common goal in almost all industrial machine-based quality control is to reach a minimal sorting error rate with a high speed.

1.3 Methods

The goal is to seek effective methods that can detect different types of wood defects and subsequently classify them. The methods are then implemented into a Matlab Program. The methods used to reach this goal are found from a literature survey in scientific areas of image analysis, machine learning and image processing.

In this thesis a supervised classifier is used by first training it with a boosted learning algorithm, Adaboost, and some image analysis methods are combined together to extract features and patterns. The data used in the training is retrieved from images of wood boards that contain the wood defects.

The detection program should be able to classify small objects that can be seen only with some few pixels such as cracks and also detect defects that cover large areas such as wood decay and blue stains.

1.4 Problem description

The task in this thesis is to implement a program for a commercial sorting machine. It is a three stage sorting machine and all boards pass through each stage on a conveyor belt. The first stage measures the density of the board and the second stage measures the geometrical defects. The third stage is a visual inspection of each board and this is where this work will be implemented.

The program should visually recognize board defects in the same manner as a sorter does. Typically, when looking at rot and fungus infections, a human recognizes this type of defects when the color changes on the board to something that is associated with the defect. The colors are not always distinctive so the sorter might look at other properties that can be found and related with the defect just to strengthen his confidence.

A human recognizes an object by looking at the features associated with the object, an experienced person has a set of requirements that each feature must satisfy so that a confidence can be built up to give the object the right label.

However, recognitions are based on experience that the person has acquired from training and learning.

The human decision making is based on the knowledge of the distinguishing features associated with the object. These distinguishing features can be anything that can represent an object e.g. color, size, weight, shape, frequency, pattern, arrangement, etc. When we find these features we can naturally decide what object we are dealing with from experience with similar objects. Object recognition in computer science requires a translation of these features into a numerical form. These feature values will then be processed in a decision making program which is called the classifier. The main purpose of a classifier is to associate each feature sample with a class label.

The defects in this work are divided into two groups. This is mainly because of how we recognize these defects. The defects that often cover large areas are instinctively detected by color or if a characteristic shape is found. Wood decay and Blue stain falls into one group because both of them miss a characteristic shape. They can take any form on the board surface, sometimes they appear as large spots or patches. The shakes and other small defects e.g. insect attacks fall into one group because they are visible with only a small amount of pixels and we can recognize them from their distinguishable shapes. These two groups will have separate approach in the recognition process.

- Large area defects:
  - Blue stain
  - Wood decay (rot)
    - False alarm objects that can be confused with rot:
      - Annual growth rings
      - Water stain
- Small defects with few pixels:
  - Shake (crack)
  - Insect attacks

In the first group one of the defects is Blue stain, the defect can appear as a patch on the board surface in any color ranging from blue to gray. The first reasonable feature that anyone can think of when recognizing the defect is its colors. Hence, working with images, colors are the central key feature in any problem. The main information source that will be analyzed is the signals retrieved from the CCD camera. The signals will be presented from the three color channels red, green and blue to form a RGB color image. The small illumination changes that can occur when a new light source is added or removed can lead to new color information. This can be misleading and increase the difficulty when evaluating the color feature of a defect. The boards might also have a small difference in color from one another.

2 Defects

When wood is used in a building as support beams, it's important to have a strong and good quality wood. If it contains a lot of knots or cracks then it can be unstable. It would be weaker in some parts and break if the weight reaches a critical level that it was actually meant to support. That is why scandinavian sawmills follow a standard regulation protocol when sorting wood. They usually follow the sorting rules by the book Nordisk Trä [FFS 1999] also known as "Blå boken" where the quality of wood boards can be divided into four classes A, B, C and D and four subclasses for A quality A1-A4. The saw mills have the possibility in many cases to increase board quality with a simple solution. That is by removing the part that contains the defect, the wood falls into a higher quality class as a result. Even if the board becomes shorter the price for it will increase. That is why it is important to not only recognize the defect but also to know exactly where it is located on the board so that the material will not be wasted.

2.1 Stain and decay

2.1.1 Blue stain

Blue stain is caused by microscopic fungi that can change the board color into blue-black and gray. The blue stain has no effect on the board strength but the condition is ideal for wood decay. The first property that one can use to detect the blue stain is its dark-blue color. The color can change with the organism and moisture content of the wood. Usually, normal wood has a moisture content between 8-13% but when wood is exposed to water and the moisture content of the wood is greater than 20%, the fungi becomes visible with the dark blue color. The blue stain does not affect the strength of the board but it's a pre-stage of other more damaging infections. The fungus spreads only on sapwood of trees which is the outer layers of the tree. There is no default characteristic shape it can appear as spots, streaks and patches along the fiber lines and sometimes it can cover the entire board surface.

Figure 1 Blue stain is the blue-gray area, sometime the blue stain can completely cover the board surface and can have varying intensities

The best way of detecting blue stain is to search for its unique color. [Knaebe and Mark 2002], [Forest Products Laboratory 1999]

2.1.2 Wood decay (Rot)

Rot is caused by fungi that decompose the wood substance. The fungus spreads within the tree both in heartwood and sapwood [Forest Products Laboratory 1999]. The part in the tree that contains the fungus is weakened so the tree could not be used as construction material according to the quality grading rules.

2.1.2.1 Problem

It is very difficult to identify rot because it doesn't have so many characteristic properties. E.g. an experienced sorter has a big problem in some cases when he evaluates a brown colored patch on timber. The brown colored patch can either be a water stain or rot. Water can cause a superficial stain on the board that has the exact same color and shape as rot, especially when rot has a faint appearance.

One of the biggest challenges in a visual based quality sorting of timber is to recognize rot. A sorter must be able to distinguish between wood decay and superficial stain. This is so difficult in some cases so that the only way to confirm if it is rot is to dig into the material and feel if it has a soft or solid consistency. Most people that work with sorting cannot describe how they recognize rot but a well established rot can be recognized easily, see Figure 2.

2.1.2.2 The extra color feature

Rot can sometimes appear with a distinctive dark-gray color seen as stripes spread along the rot, see Figure 2. The gray color is not always visible even if rot is well established so a classifier based on this feature is not entirely reliable.

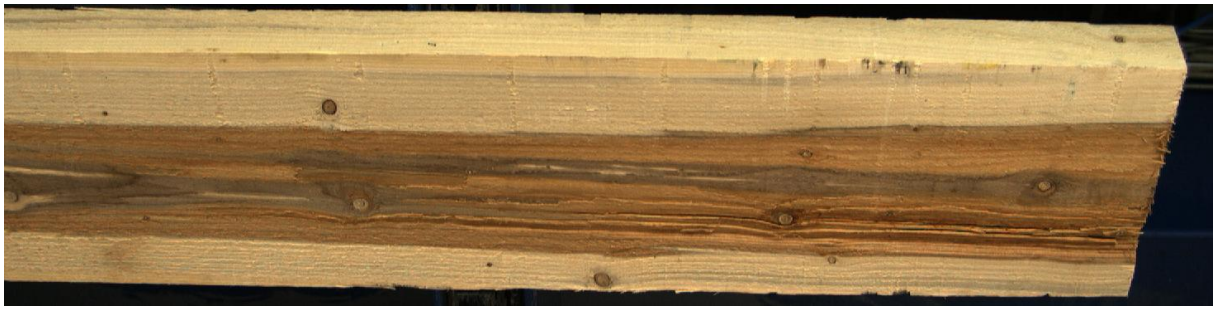


Figure 2 A well-established rot that is very distinct with its gray stripes

In Figure 2, the rot is very distinct and anyone can see that this is not a healthy board. The color of rot can be very similar to other objects which is commonly found on the board e.g. branches, bark, annual growth rings and compression wood.

This limits an identification of rot that is based entirely on color properties so to recognize rot a detector should be able to identify at least some other properties e.g. the shape or texture features. The first idea to solve this problem was to use the same technique as in [Viola and Jones 2001]. But this will not work because the shape of rot cannot be predicted and the texture pattern is more or less similar to the rest of the board.

2.1.2.3 False objects

Some samples of rot can be seen in Figure 3. Samples of superficial stains and common shapes found on a board, which can be mistaken for rot, can be seen in Figure 4.

Figure 3 Rot appears with various brown colors and the intensity is also changing within the rot, it also misses a default shape. The gray color, within rot, can be seen on the board to the far right but it's not always clearly visible.

Figure 4 These examples show some boards that can be mistakably identified as rot. Patches created by water stains can have the same color and form as rot. On the first board seen from the left, the annual growth rings (dark-brown) are usually thin lines but in many cases they appear as large patches that have the form of uneven waves with a repeating pattern along the board. Water stains can easily be recognized when they appear as in the image on the far right. The horizontal edge is formed by a stick which was covering the board during a drying treatment and then removed when sorting the board.

The rot defect is a fuzzy problem which can be mistaken for anything that has similar color and a random shape. The only option for visually recognizing rot is to separate rot from everything else that actually has a characteristic property. This is an indirect search for rot by excluding any false object that has the same color as rot but also has a recognizable property.

2.1.2.3.1 Water stain

There is one way to distinguish rot and water stain. The image on the far right in Figure 4, shows a pattern for water stain that can be exploited. It is clearly seen that the brown color is cut of horizontally because of a stick that was covering that area during a drying treatment. This distinguishing feature can be used in the recognition process. The horizontal line can be detected by using a line detecting method, Hough transform [Duda and Hart 1972], on an edge image. If this pattern appears and if it can be extracted from the image then it can be used to indicate that the object which is being analyzed has a high probability of being a false alarm (water stain).

2.1.2.3.2 Annual growth rings

The wave shaped growth rings appear often because the tree geometry is not perfectly cylindrical. The growth rings can sometimes look like large brown wavelike objects on the board surface. The waves slowly rise and fall sharply in tone in a repeating pattern but each wave is unique and they change in size and distance from one another in an unpredictable way. These annual growth rings have a distinguishable shape and pattern on the board which can easily be recognized, see Figure 5. If the search for rot is entirely based on color detection then one would find the annual growth rings as detected rot because of their similar color. That is why it is important to detect annual growth rings to warn for false alarms when searching for rot.

Figure 5 A typical sample of how annual growth rings could look like on the surface of a board. The dark brown color is similar to the color of decayed wood.

2.2 Shake

The typical shape of a large crack on a board is long and narrow. This makes the detection of shakes almost impossible if the same methods are used as in rot and blue stain detection. The cracks are so narrow so that it can be a problem to capture the details if the camera has a low image resolution. It will be assumed that the image details are high enough so that one can use classical image analysis tool such as Hough transform [Duda and Hart 1972], to detect lines in combination with an edge detector. SUSAN will be used to extract edges from an image.

2.2.1.1 False objects

The biggest challenge is not to detect the lines on the board. There are other objects that are similar to shakes. One of them is the pith which can appear on the surface when the board is cut out from the tree's central part. The pith is a long and narrow straight line on the surface of the board, see Figure 6.

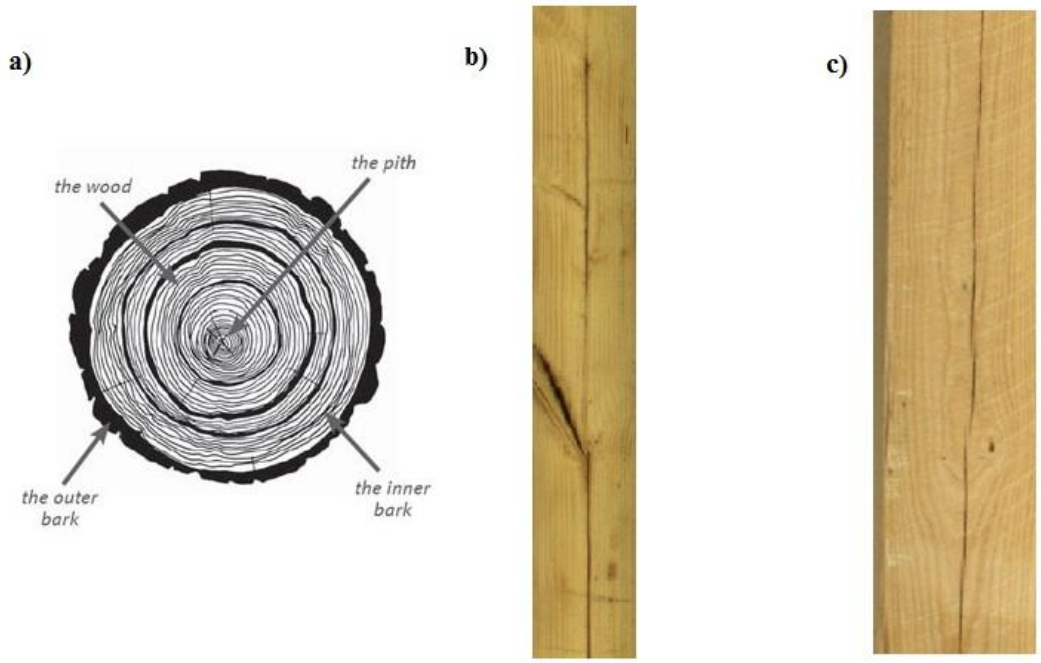


Figure 6 The pith is the central annual growth ring which can be 1-2 cm in diameter when looking at the cross section of a tree trunk, the color is dark brown a). When a board is cut out from the central part of the tree, close to the pith, a long and narrow dark line can be seen on the board surface b). A typical pattern of how crack may appear c). Image a) ref. www.pir.sa.gov.au

When looking at a board it is easy to see the difference between a shake and a pith line. The pith line is seen some times as a straight line that goes through the board from bottom to the top. The form of the pith and the tree is not perfectly straight so the line can appear and disappear along the board. The pith color is dark brown and this is a property that can be used to easily distinguish between shake and pith. A detection based on the color property requires a high resolution image with a scene that has a good ambient light so that the true colors can be captured otherwise both the shake and the pith will be smeared out. There is always a loss when sampling real signals into digital values.

When looking closely at a shake one can see a pattern that is difficult to see from a distance. The shake is a discontinuous line that sometimes can be seen

as parallel lines, the pith is never parallel because it's only made of one line, and this fact can be used if this pattern is detected.

Figure 7 Sometimes a crack can have a discontinuous pattern which is not seen on a pith line.

The shake is often a curvaceous line compared to the pith. The fact that it is curvy is not enough to state that the object on the image is a shake because a shake can be very straight in some cases but by tracking a line one can analyze the derivative of the line and its directional changes. This fact can help to strengthen the confidence of a decision making program.

There is another good fact that can be used to detect the pith. In reality it is not that difficult to recognize a shake and pith because the sorter sees the whole board when evaluating it.

If a pith line exists then it is clearly seen as a long straight line appearing with unpredictable interval along the board, the board can be up to 10 m long, it also has branches growing from it which can be seen as a dark patch in Figure 6 b).

3 Background Theories

3.1 Colors

Isaac Newton showed in the 17th century that white light from the sun is a mixture of colors by splitting it with an optical prism. The visible light is only a small part of the electromagnetic spectrum with a range of wavelengths from 380 nm to 740 nm.

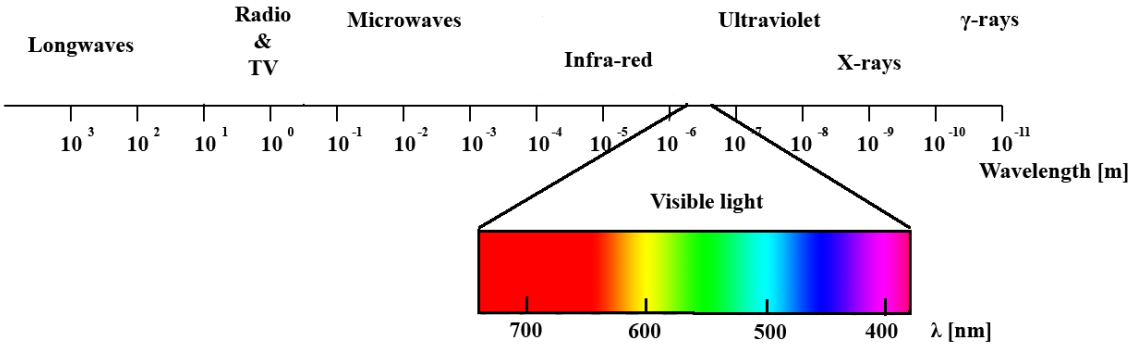


Figure 8 The Electromagnetic spectrum. Our eyes can only sense a small range of wavelengths, this small window in the EM spectrum is called visible spectrum.

The color of an object is related to the structure of the objects material. When light reaches the surface of the object it will be reflected or absorbed at different wavelengths depending on the composition and microstructure of the objects material. E.g. the color of plants is green because all the colors in the white light from the sun are absorbed by the plant except the green light which is reflected into our eyes.

The human is able to see colors thanks to the inner surface of the eye which contains sensitive photoreceptor cells. There are essentially two types of cells the rods and the cones. The rods provide us with black-and-white vision, when the light intensity level is low e.g. during the night. The cones provide us with colored vision during daytime. The cones are divided into three groups based on their property of absorbing light in different wavelengths: the short (S) with its maximum sensitivity at 430 nm, medium (M) at 560 nm and long (L) at 610 nm [Kandel et al. 2000].

A combination of these cones can give the human the ability to sense all the colors from the visible spectrum, see Figure 8. The same principles are often used when displaying an image on a computer or a TV screen.

3.1.1 RGB color model

The RGB color model has its origin in color television where the three primaries red, green and blue was used to mimic the phosphor in the cathode ray tube (CRT).

Usually, on a TV screen, one pixel contains three light sources which each and one of them emits the colors red, green and blue (RGB) in an additive mixture to display any color, see Figure 9.

In vector form, the white color has the RGB value (1, 1, 1) if (R, G, B) ∈ [0, 1]. The color gray is the value (0.5, 0.5, 0.5) and any shade of gray can be chosen with (X, X, X) if X ∈ [0, 1]. If one would like to change the intensity of a color, let's say red (1, 0, 0), then one could do that simply by multiplying the (R, G, B) vector with a scalar. However, if we would like to change red to a more saturated red we would have to change all three parameters to have the right mixture.

3.1.2 The RGB to HSV color transformation

With a HSV color model one can easily select colors with a Hue, saturation and value parameters. The solution to the unintuitive way of selecting colors in the RGB model (Cartesian coordinate system) is to transform it into a cylindrical coordinate system, HSV. The angular component is the Hue which represents the pure color. The radial component controls the saturation and the height component changes the value from 0 (black) to 1 (full color). The transformation is illustrated in Figure 10.

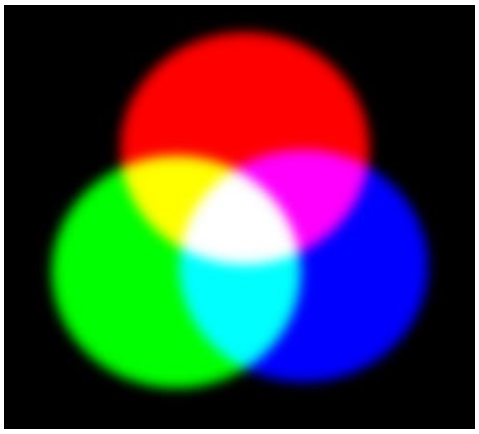


Figure 9 An additive color mixture can give the human the ability to sense other colors than the primary ones used in the mixture e.g. red, green and blue (RGB color model).

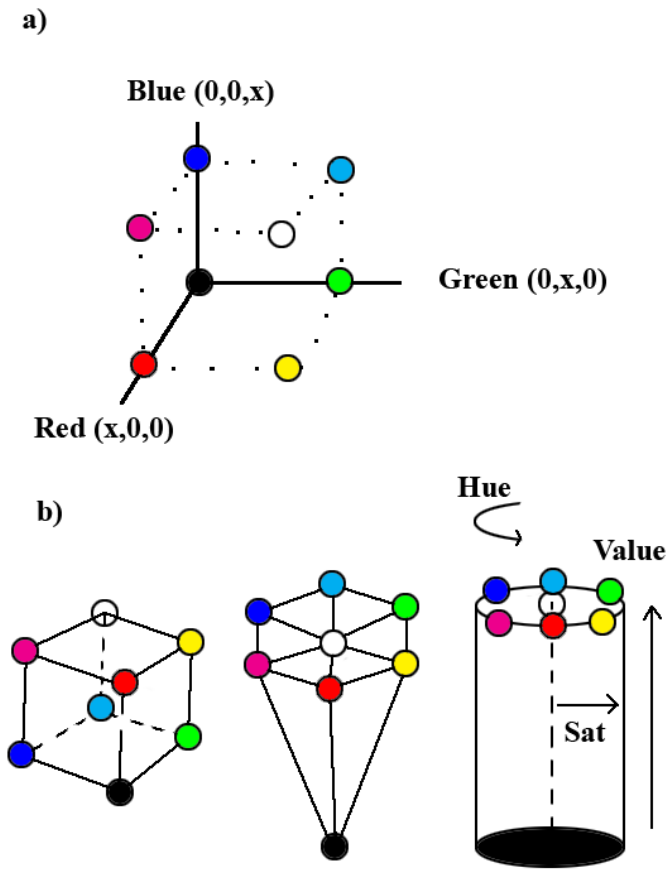


Figure 10 In a), the three color primaries red, green and blue are the orthogonal axes that span the RGB color space. The geometrical transformation from the RGB color model to HSV is illustrated in three steps starting from the left. Instead of choosing the right amount of red, green and blue to produce a color with the right concentration one can easily pick the pure color with the Hue component and change the intensity with the value component.

The HSV color model can be seen as a geometrical transformation of the RGB color space. The RGB to HSV color transform algorithm is given in *Table 1*.

Table 1 RGB to HSV algorithm [Smith 1978].

Input: R, G, B ∈ [0,1]

1 $V \coloneqq \max(R, G, B)$

2 $X \coloneqq \min(\mathrm{R}, \mathrm{G}, \mathrm{B})$

3 $S \coloneqq \frac{V-X}{V}$, if S=0 return;

4 $r \coloneqq \frac{V-R}{V-X}$, $g \coloneqq \frac{V-G}{V-X}$, $b \coloneqq \frac{V-B}{V-X}$

5 if $R = V$ then $H \coloneqq (\text{if } G = X \text{ then } 5 + b \text{ else } 1 - g)$
if $G = V$ then $H \coloneqq (\text{if } B = X \text{ then } 1 + r \text{ else } 3 - b)$
else $H \coloneqq (\text{if } R = X \text{ then } 3 + g \text{ else } 5 - r)$

6 $H \coloneqq \frac{H}{6}$

Output: H,S,V ∈ [0,1] where H=0 is the red color.

More information about color models and transformations can be found in [Sonka et al. 2007].

3.2 Classifier

3.2.1 Basic concepts

When faced with a problem on how to label some objects, it is important to be able to extract information about the object properties. These features will be used to discriminate all the objects from one another. If the objects share the same properties then they can be grouped together with the same label.

E.g. A typical situation is when a person is visiting a recycling center. The task for that person is to dump the trash in the right container, so he follows the labels of each container; metal, glass, plastic, paper, etc. With his experience he already knows what metal, glass, etc. is like. With a strong confidence he can associate a tin can with metal by only looking at its shape because he already knows that a tin can is made of metal.

By analogy, the person is the classifier that is making all the decisions and therefore labeling the objects. The objects are samples and with each sample some features or properties are given.

A feature can be divided in two types, the quantitative features are typically numerical that is discreet or continuous values while qualitative features are categorical. Statistical pattern recognition is when the feature type is quantitative, e.g. a gray-level intensity of a pixel, the weight of a person, the time it takes to run a mile, etc. A sample *s* with *n* features is presented as an n-dimensional vector, $\boldsymbol{x} = [x_1, \ldots, x_n]$, $\boldsymbol{x} \in \mathbb{R}^n$ where $\mathbb{R}^n$ is a real space called the feature space.

There are two categories of classifiers: the unsupervised and supervised learning classifier. The unsupervised learning is a classifier that has the capability of grouping a set of samples by clustering their feature values. This is useful when one is interested in knowing if there are any groups in the data. The classifier analyzes the data structure and finds out if there is any similarity among the samples to classify them into undefined classes.

The supervised learning is when a classifier is trained on a data set, which has a class label l ($\boldsymbol{l} = [l_1, \ldots, l_n]$) assigned to each sample, so that it can predict the class labels of unseen samples. The classifier is provided with a learning skill which is used in the training. Usually the data set is divided into two groups the training set and the test set. This is to measure how effective the learning was during the training. The test set will not affect the learning of the classifier because it will be used for measuring how the classifier performs on unseen samples. A quantitative classifier is a function, $\boldsymbol{f}$, that transforms any input value $\boldsymbol{x}$ into an output $\boldsymbol{l}$ [Kuncheva 2004].

$$f: \boldsymbol{x} \rightarrow \boldsymbol{l}, \boldsymbol{x} \in \mathbb{R}^n \tag{1}$$

E.g. In a two-class problem, Figure 11, where $\boldsymbol{l}$ = {-1, 1}, a linear classifier could look like this,

$$f(\boldsymbol{x}) = 10x_1 + 3x_2 - 10 = 0 \tag{2}$$

If the value of a vector $\boldsymbol{x}$ is not the solution of *eq. (2)* then the outcome of *f* will be a positive or a negative value and $\boldsymbol{x}$ will be assigned to respective label, $\boldsymbol{l}$. The classifier's decision border which is made up out of a linear equation divides the feature space $\mathbb{R}^2$ in two regions and any sample value found in one of the regions belongs to the class of that region.

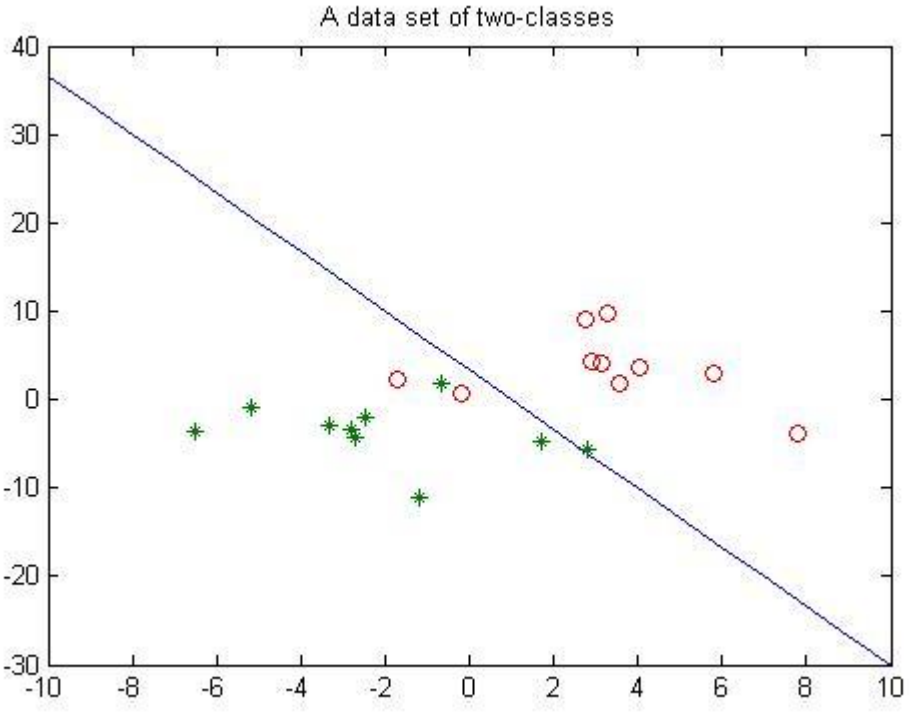


Figure 11: Example of a classifier from eq. (2) on data set of two classes.

The performance of the classifier is measured with an error rate. The error rate is the amount of misclassified samples, N_{error}, divided by the total number of samples, N_{tot}.

$$error = \frac{N_{error}}{N_{tot}} = \frac{\sum_{i=1}^{n}[I(f(x_i), l_i)]}{N_{tot}} \tag{3}$$

Where,

$$I(f(x), l) = \begin{cases} 1, & f(x) \neq l \\ 0, & f(x) = l \end{cases} \tag{4}$$

In *figure 6,* the amount of samples is 20 and with the discriminating classifier from *eq. (2)* the amount of misclassified samples is 3, the error rate would simply be $3/20$.

3.2.2 Adaboost

The Adaboost was first introduced by [Freund and Schapire 1995]. They showed that with the Adaboost algorithm one can train any weak learning algorithm into a strong classifier, f, which is an ensemble of the weak classifiers.

$$f(X) = \sum_{t=1}^{T} \alpha_t h_t(X) \tag{5}$$

Where h is a hypothesis or the learner that is weighted with a coefficient α, α is calculated in Adaboost as a consequence of choosing h. [Meir and Rätsch 2003]

To understand how Adaboost works I refer to the on-line allocation model which is formalized by [Freund and Schapire 1995] in this way, if A is an allocation agent and there is N numbered options or strategies which A can choose from, then at each time step (t= 1, …, T) A decides on $\boldsymbol{P^t}$, which is an allocation distribution over the strategies. $P_i^t \geq 0$ is the amount allocated to strategy i,

$$\sum_{i=1}^{N} P_i^t = 1 \tag{6}$$

Each strategy suffers some loss l_i^t that is influenced by the environment. The loss suffered by A is then $\sum_{i=1}^{N} P_i^t \, l_i^t = \boldsymbol{P^t l^t}$, the loss is bound to $l_i^t \in [0,1]$.

The goal of allocation algorithm A is to minimize a net loss. It tries to minimize its own cumulative loss relative to the cumulative loss suffered by the best strategy.

$$\boldsymbol{L}_A - \min_i \boldsymbol{L}_i \tag{7}$$

Where,

$$\boldsymbol{L}_A = \sum_{\boldsymbol{t=1}}^{\boldsymbol{T}} \boldsymbol{P^t l^t} \tag{8}$$

, is the total cumulative loss suffered by A up to time step T and

$$\boldsymbol{L}_i = \sum_{\boldsymbol{t=1}}^{\boldsymbol{T}} \boldsymbol{l_i^t} \tag{9}$$

, is the total cumulative loss of strategy i. In other words, the allocation algorithm A has to find a way of using many strategies together so that it can be more accurate than the best strategy. Adaboost is designed to handle this kind of problem.

In [Freund and Schapire 1995], they described with game theory that it is difficult to predict an outcome perfectly with one rule so they showed that with a set of rough "rules of thumb" one could use all rules in a linear combination, with a weight on each rule that depends on its correctness, and

turn them into one strong rule. E.g. a stock trader would follow a set of trading rules to try to predict the stock value, from experience, some rules works better than the rest, however, he still uses all of them together but he puts more attention on the favorite rules.

In analogy, Adaboost builds a strong classifier *f* incrementally from a training where a weak learning algorithm chooses *h* in each boosting round from a set of hypothesis that are tested on a set of labeled samples; : $\{(X_1, Y_1) \ldots (X_N, Y_N)\}$, $X \in \mathbb{R}^n$ with their labels, e.g. in a 2-class problem $Y \in \{-1,1\}$.

Providing the weak learning algorithm *h* with a weight $\boldsymbol{w}$ on each sample *x*, Adaboost trains the new *h* on *x* in a way that the previous misclassifications made by *f* is taken into account when evaluating *h*. $\boldsymbol{w}$ is calculated out of the error rate made by the previous total strong classifier *f*. One could say that $\boldsymbol{w}$ is the link between the previous boosting rounds and the present boosting round. The weight vector is updated according to

$$w_i^{t+1} = \frac{D_t(i)e^{-\alpha_t h_t(x_i)y_i}}{Z_t} \tag{10}$$

Where, $D_t(i)$ is the weight distribution over the sample i in the present round and Z_t is a normalization factor so that w_i^{t+1} will be a normalized weight distribution in next round.

$$Z_t = \sum_{i=1}^{N} D_t(i)e^{-\alpha_t h_t(x_i)y_i} \tag{11}$$

The weight associated with an example is increased if the example is hard. The difference between Adaboost and previous algorithms is that it can adjust adaptively to the error rate of the hypothesis returned from the weaklearner. This is why it was named Adaboost. The error rate of a hypothesis at time step *t* is:

$$\varepsilon_t = \sum_{i=1}^{N} w_i^t \, I(h_t(x_i), y_i) \tag{12}$$

The selected *h* will then be added to the ensemble and its influence on the total outcome is weighted by the coefficient α. The weight α and base classifier *h* is chosen in each round *t* in a way so that they minimize eq. (11) in a greedy way, [Schapire 2001], [Schapire and Singer 1999]. Note that eq. (11) is the loss, eq. (8), as described in the on-line allocation model. For a binary problem, this is done optimally by having α:

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) \tag{13}$$

Table 2 Adaboost training.

The Adaboost algorithm for a two-class problem.

Input: N labeled samples $S = \langle (x_1, y_1) \ldots (x_N, y_N) \rangle$ where $x_i \in \mathbb{R}$ and $y_i \in \{-1,1\}$. $i = 1, \ldots, N$

Initialize: the weight distribution over the samples $w_i^t = D(i)$ (usually in the first round, $w_i^1 = D(i) = \frac{1}{N}$).
Set the number of boosting rounds T.

FOR t=1, …, T

1. Provide the **weaklearner** with weight vector $\boldsymbol{w}^t$ and S
   Get back a hypothesis $h_t: x \to [-1,1]$
2. Calculate the error rate of h_t with *eq. (12)*, and return h_t with the lowest error rate.
   $\varepsilon_t = \sum_{i=1}^{N} w_i^t \, I(h_t(x_i), y_i)$
   I is described in *eq. (4)*
3. Stop If $\varepsilon_t > 0.5$
4. In a binary problem Set, see [Schapire 2001] and [Schapire and Singer 1999]. Use *eq. (13)*.
   $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$
5. Calculate the new weight distribution, use *eq. (10)* & *(11)*.
   $w_i^{t+1} = D(i) e^{-\alpha_t h_t(x_i) y_i}$
   and normalize,
   $\boldsymbol{w}^{t+1} = \frac{\boldsymbol{w}^{t+1}}{\sum_{i=1}^{N} w_i^{t+1}}$

Output: final classifier
$f(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

The only requirement of the base classifier h is that it should perform better than random classification so the error rate of the weaklearner should be lower than 50%.

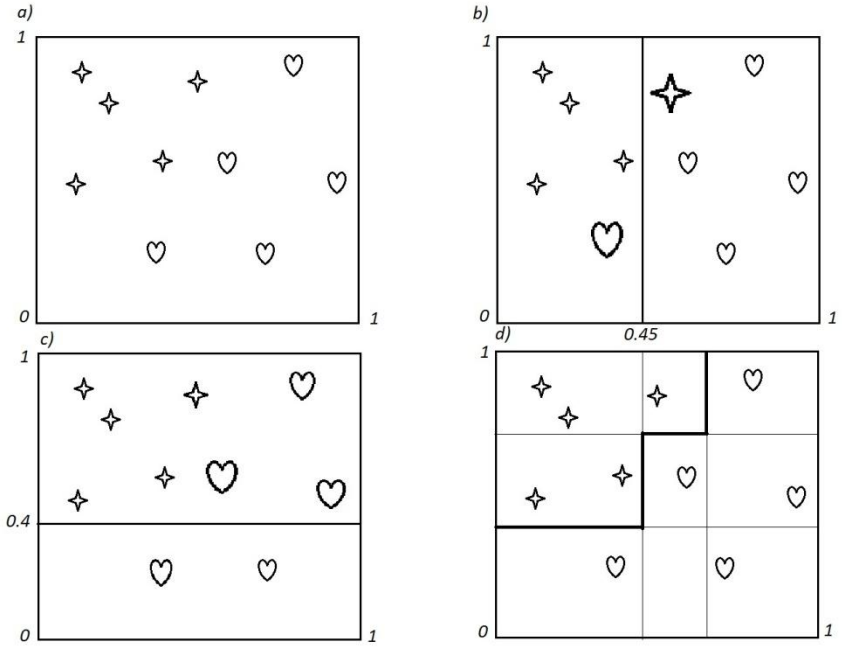


Figure 12 a) A two-class sample set is used in Adaboost training with equally distributed weight among the samples. b) Adaboost trains the weaklearner, which is a threshold in this illustration, to separate the two classes with minimum error rate. The weight distribution is recalculated from the error rate and then redistributed among the samples, according to step 5 in Table 2*. In this illustration, the size of each sample icon is proportional to the weight given to it. A large icon illustrates a large weight because of the misclassification made by the Weaklearner in previous round. c), is the next boosting round in this training. d), Illustrates a decision border that is made up of all hypotheses weighted together, the bold line is the decision boundary of the final classifier.*

3.2.3 Error handling

Adaboost trains on two sample sets with the weaklearner. If N is the total amount of the available data sample, then one third of N will be used as the training set and leaving two thirds of N as the validation set. The training error rate is minimized by Adaboost at each boosting round and if the samples from the two classes are not mixed together too much then the training error rate can reach a zero level. It is proven by [Freund and Schapire 1995] that the training error of the final classifier drops exponentially fast as T increases. The final classifier has the error $\varepsilon = Pr_{i\sim D}[f(x_i) \neq y_i]$ which is bounded above by:

$$\varepsilon \leq e^{-2\Sigma_{t=1}^{T}\gamma_t^2} \tag{14}$$

Where $\gamma_t = \frac{1}{2} - \varepsilon_t$, ε_t is the error of hypothesis t and $\gamma_t > 0$. To simplify eq. (14), let say that all hypotheses have the same error rate so that $\gamma_t > \gamma > 0$:

$$\varepsilon \leq e^{-2T\gamma^2} \tag{15}$$

Eq. (15) shows us that when T increases the upper bound of the error rate of the final classifier is lowered exponentially.

While the training is carried out, the classifier is evaluated at each boosting round with the validation samples.

The validation error rate is heavily influenced by the size of N during the training.

3.2.3.1 When N is small

When N is small, that is when we have few sample data available, the test error rate and training error rate follows a trend at opposite direction.

Adaboost minimizes the training error by separating the samples from the two classes as accurate as possible, it draws its decision boundary to fit the small amount of data sample, but the problem is that the boundary is not suitable for the real sample structure. The small amount of data samples gives a fuzzy separation boundary. That is why the validation data does not fall into the regions that Adaboost finds and the test error rate is not minimized as effectively as the training error rate.

3.2.3.2 When N is large

In contrary, if N is large, the training data would be dens and the boundary that separates the two classes would be more obvious but if the two classes are mixed together to a certain extent then a precise separation would be more difficult to achieve. Adaboost might separate the training samples perfectly but it is over-fitting the final classifier and the validation error rate increases instead.

The training error rate and the validation error rate would more or less follow the same trend directions when N is large and the risk of over-fitting the classifier is lower.

3.2.3.3 Moving the boundary

The lowest error rate that can be reached depends solely on how well the sample values are separated. If one is interested in a classifier that has a higher accuracy in classifying class 1 samples than class 2 samples then one can increase the amount of class 1 samples in training mode so that they outnumber class 2. This will force the weaklearner to move its decision boundary to include more of class 1 as a result.

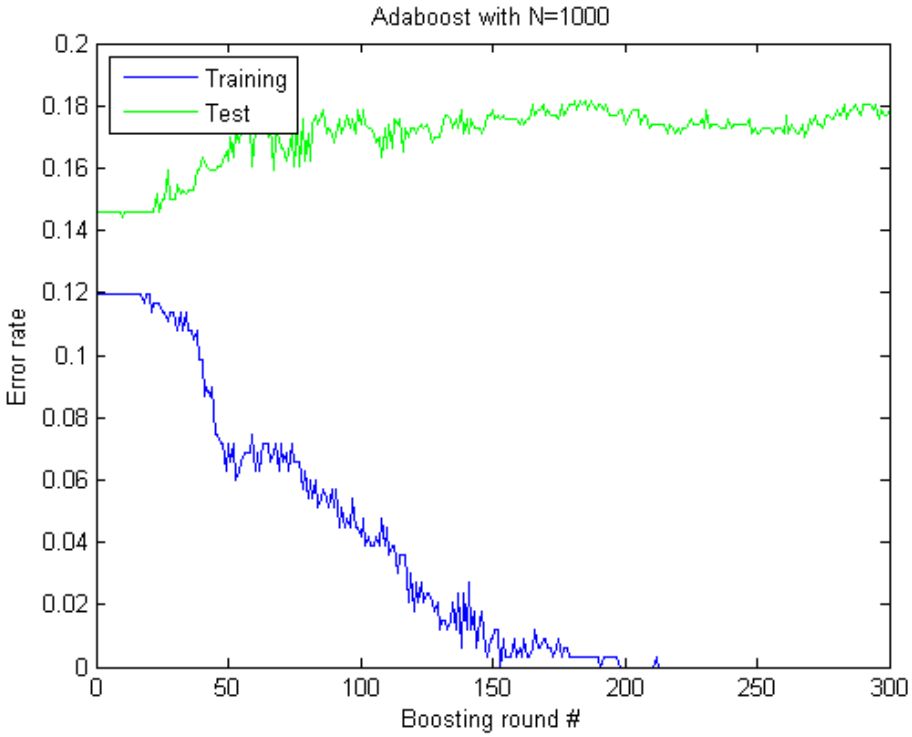


Figure 13 The amount of samples (N=1000) used in training the classifier is low. The test error shows that the classifier is over fitting almost directly at the beginning. Even if it successfully has driven down the training error to 0% the classifier is poor in predicting the real test data. It's only 18% accurate.

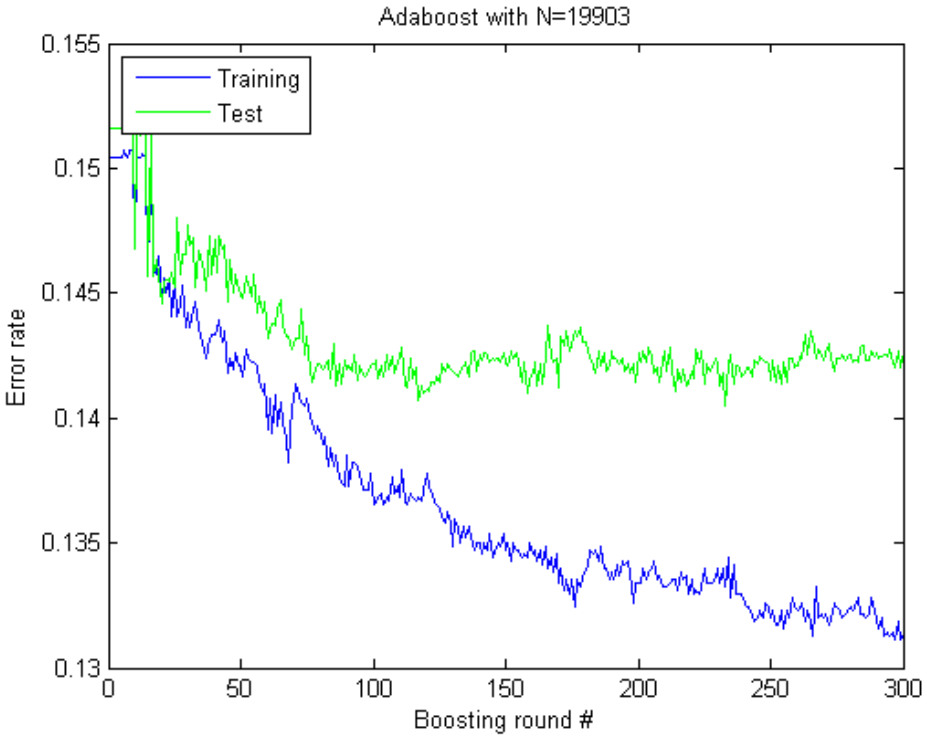


Figure 14 more training samples are available (N=20000). The test error trend follows the training error at the beginning but at 80 boosting rounds the classifier is not

improved in predicting the test samples. The large amount of data available shows the classifier clearly where the two classes are located in the feature space but the samples are also mixed together at a higher rate at the boundary where they meet which makes it difficult for Adaboost to further improve the error rate.

3.2.3.4 Cross-validation

The large amount of training data used to find the true defect colors makes it possible to train Adaboost by dividing the training data into sub groups and run the training and validation in each group to find the optimal classifier.

The final classifier is then tested on all available samples and the validation error result will then be used to rate the final classifier.

3.2.3.5 Multiple classifiers

The rating will be used as a weight for the final classifier to indicate how reliable it is. Adaboost will train classifiers with different color models RGB, Lab, HSV, etc. to search for a defect's color. Each color space classifier will have different accuracy when predicting a pixel sample. By using the performance rating one could use all final classifier at the same time eq. (16). The best classifiers have the lowest error rate and they will have a higher importance in the ensemble, almost in the same way when Adaboost weights the hypothesis h_t with α_t.

The final classifier *f* can be combined together with

$$G(x_i) = \frac{\sum_{K=1}^{M} f_K(x_i)(1 - err_K)}{\sum_{K=1}^{M}(1 - err_K)} \tag{16}$$

Where, $G(x_i)$ is the final output from a pool of M final classifiers. $G(x_i) \in [0,1] \because f_K(x_i) \in \{0,1\}$. err_K is the test error rate of the final classifier f_K. One classifier based on one color model works fine but the prediction from a group of classifiers from different color models can increase the accuracy.

3.2.4 Weaklearner

Any classifier could be used as a base classifier and trained by the Adaboost algorithm as long as they perform better than random classification. The weaklearner used to predict the color of a defect in this thesis is mainly a function that can separate the feature space $\mathbb{R}^2$ and $\mathbb{R}^3$ into two regions. This can be done in many ways e.g. with circles, lines, planes, spheres, etc. Adaboost tries to drive down the training error rate on all weaklearners without over fitting but the speed of convergence of the training will be different. Some weaklearners will push down their training error faster, with fewer boosting rounds.

A weaklearners purpose in each boosting round is to return a hypothesis, *h*, that will be selected from a set of available hypothesis in the weaklearner, where $Hypothesis: \{h_1, \dots, h_m\}$. Providing the weaklearner with labeled samples, $\boldsymbol{S}$, and their weight distribution, $\boldsymbol{w}^t$, the weaklearner should return from the set *Hypothesis* an h_t that minimizes the training error ε_t:

$$\underset{h_t \in Hypothesis}{\operatorname{argmin}} \left\{ \varepsilon_t = \sum_{i=1}^{N} w_i^t I(h_t(x_i), y_i) \right\} \tag{17}$$

, *I* is described in eq. (4).

3.2.4.1 Binary decision stumps in multiple dimensions

The simplest way of separating $\mathbb{R}^2$ or $\mathbb{R}^3$ space is to draw a line or a plane. The weaklearner, decision stumps, is a program that creates a set of *hypothesis*. A hypothesis in the set is simply a plane in $\mathbb{R}^3$ that slices the volume that the training samples where spanning.

A plane (in 3D) or a line (in 2D) is selected that will minimize the error rate according to *eq. (*17*)*. In this weaklearner, the hypothesis are simply thresholds in (X, Y, Z) Cartesian coordinate system that divides the space into two regions.

In a 3D feature space, the weaklearner will look for the range of the feature space by finding the minimum and maximum value of samples in x-coordinate, y-coordinate and z-coordinate. This range of the data set is then

$$\begin{aligned} &\text{X-Coordinate: } R_x \in [\min(S_X), \max(S_X)] \\ &\text{Y-Coordinate: } R_y \in [\min(S_Y), \max(S_Y)] \\ &\text{Z-Coordinate: } R_z \in [\min(S_Z), \max(S_Z)] \end{aligned} \tag{18}$$

Where, *S* is a sample with feature values in *x*, *y* and *z*. The hypothesis set contains thresholds that divide the ranges to a number of equally spaced pieces (step resolution). The set is simply orthogonal planes at different positions. A plane will divide the space into class-1 region (over the plane) and class-2 region (under the plane) or vice versa and then tested with *eq. (*17*)*. When the weaklearner finds the threshold that has the minimal error rate, let's say that this threshold is x_l, then it will focus on the interval between $[x_{l-1}, x_{l+1}]$ and search for a new threshold with a finer resolution. It will divide the interval with the same step resolution. The number of iterative refinements that should be done is decided in advance.

The weaklearner will then return one of the planes or lines, which is given as a threshold in one of the coordinates, and also which side the class-1 region is relative to the plane.

The performance of this type of weaklearner depends on the step resolution. If the Hypothesis set contains few planes then the weaklearner might not find

the optimal threshold to separate the two classes from each other so it is recommended to have a high step resolution. Put another way, a small spacing between the planes so that a higher precision can be achieved within each boosting round.

Table 3 Inside the weaklearner decision stumps.

Input: Weight vector $\boldsymbol{w}$ for N labeled samples $S = \{(\mathbf{v}_1, \mathrm{y}_1), \ldots, (\mathbf{v}_\mathrm{N}, \mathrm{y}_\mathrm{N})\}$ where $\mathbf{v}_\mathrm{n} \in \mathbb{R}^3$ and $y_i \in \{-1,1\}$. $n = 1, \ldots, N$, *see Table 2: step 1.* Note: The only difference from table 2 is that $\mathbf{v}$ is a three dimensional feature vector.

Set: The step resolution, Δ, the depth d, a depth counter d_{count}=0.

1. Find the volume spanned by $\mathrm{R_x}$, $\mathrm{R_y}$ and $\mathrm{R_z}$, see *eq. (18)*. E.g. $R_x \in [R_{x\,\min}, R_{x\,\max}]$
2. Search for the optimal threshold on x-axis, by first dividing the interval, $R_{x\Delta} = \frac{|R_{x\,\max} - R_{x\,\min}|}{\Delta}$.
   $\mathbf{B} = \{R_{x\,\min}, R_{x\,\min} + R_{x\Delta}, \ldots, R_{x\,\max}\}$, Test all threshold values in $\boldsymbol{B}$ with the sample values $\boldsymbol{v}$.
   For i=1, ..., Δ
   For n=1, ..., N
   If $\mathrm{v_{xn}} \geq \mathbf{B}(i)$
   Then $class_i(n) = 1$
   Else $class_i(n) = -1$
   END IF
   END
   END
   $side_i = 1$ // this denotes that class 1 ($y = 1$) is over the plane.
3. Calculate the error rate similar to *eq. (17)*, $h_t(x_n) := class_i(n)$:
   $\varepsilon_i = \sum_{i=1}^{N} w_n I(class_i(n), y_n)$
   , where *I* is described in *eq. (4)*.
4. If the error rate is $\varepsilon_i > 0.5$ then change the side of the plane so that class-2 ($y = -1$) is over the plane:
   IF $\varepsilon_i > 0.5$
   $side_i = -1$, $\varepsilon_i = 1 - \varepsilon_i$
   End IF
5. b is the threshold value with the lowest error rate. Refine the search by focusing on the interval close to b.
   IF $d > d_{count}$
   $d_{count} = d_{count} + 1$
   $R_{x\,\min} = b - R_{x\Delta}$
   $R_{x\,\max} = b + R_{x\Delta}$
   Go through step 2-5 with the new $[R_{x\,\min}, R_{x\,\max}]$
   END IF
6. Through step 1-5, Do the same thing for y and z coordinates, and compare the error rate from each coordinate so that the threshold that has the lowest error rate will be selected as the final hypothesis.
7. **RETURN** $\boldsymbol{h} = \{b, \varepsilon, side, \boldsymbol{class}, xyz\}$
   where the *xyz* denotes which coordinate the threshold value belongs to.

This weaklearner will then form with Adaboost a strong classifier that will eventually separate the feature space and find the right decision boundaries where class-1 and class-2 is separated. The weaklearner slices the feature space horizontally and vertically so it is not so effective in separating the features at round surfaces.

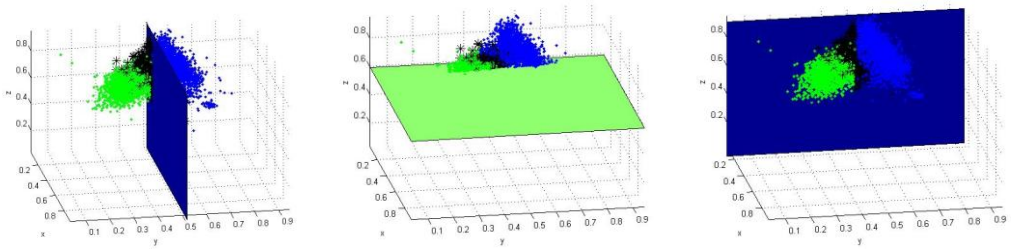


Figure 15 Example of how the weaklearner "Decision stumps" looks like when Adaboost trains on a two-class sample data. A plane is chosen successively in each boosting round and then added to the ensemble so that they will form a decision surface (or border in 2D) with the final classifier that separates the two classes. The blue and green dots represent the feature values from each class and the black dots are the misclassifications made by the hypothesis.

Even if the samples in the training set are overlapping, the weaklearner will try to separate them as good as possible as long as the separation error is lower than 50%.

3.2.4.2 Bubbles

The orthogonal planes have difficulty in forming a boundary with round corners or a smooth surface. This lead to a new type of weaklearner that is better suited for separating the regions in the feature space. Instead of having orthogonal planes a spherical shell is used to separate the feature space into two regions similar to the orthogonal planes.

At each boosting round, a number of spheres will be placed randomly within the volume that the feature samples occupy, eq. (18). In the same way when bubbles are created randomly in a soda drink. The hypothesis set is the bubbles generated at different locations with radiuses $r = \{\Delta, 2\Delta, 3\Delta, \dots, n\Delta\}$. Each sphere in *Hypothesis* is tested with *eq. (17)* and the sphere with the lowest error rate will be selected. The best sphere is the one which can separate the two classes from each other most efficiently.

The weaklearner will return a radius, the center of the sphere given in xyz-coordinates and which side class-1 belongs to (inside or outside the spherical shell).

When the final classifier is complete it will confine one of the classes within a volume or multiple volumes defined by all the spheres selected from each boosting round. The strong classifier separates the two classes from each other with a smooth decision boundary, see Figure 17. Imagine that the spheres

used in the ensemble forms a 3D geometry that resembles a model of a molecular structure of atoms with different sizes. If the feature space is only 2-dimensional then the hypothesis set is made by circles instead of spheres.

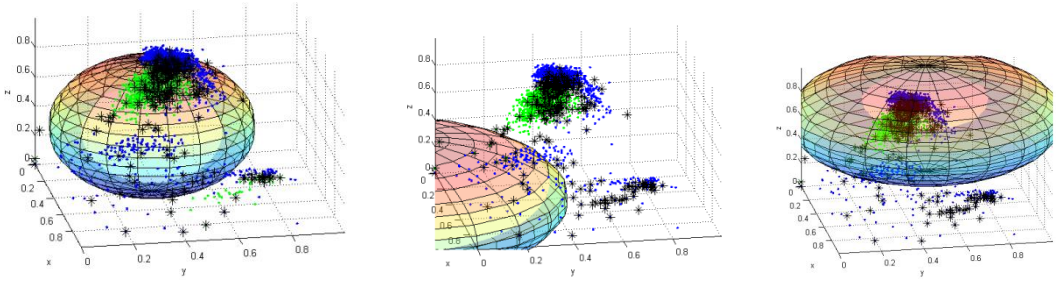


Figure 16 Adaboost trains the weaklearner "Bubbles" with a two-class sample set, green and blue. At each boosting round a set of spheres are generated at random locations. Each sphere is tested with different radiuses to find the best way in separating the sample data. The black dot is the misclassified samples. The strong classifier will be a volume defined by all spheres, see Figure 17.

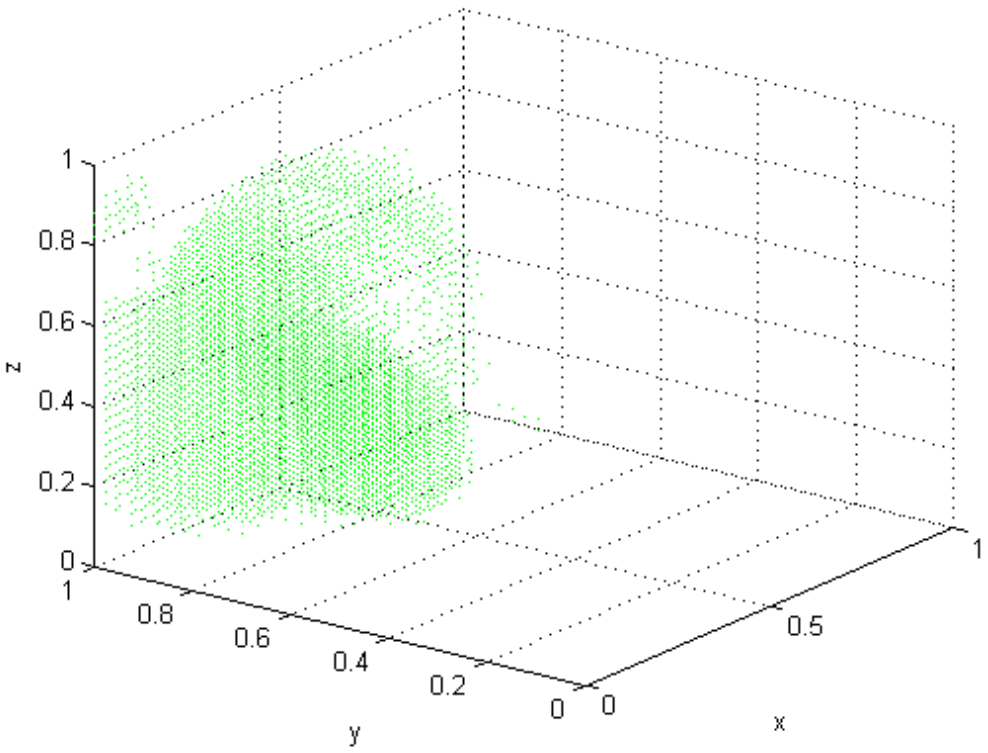


Figure 17 The green cloud represents the volume which is confined within decision surface which is made up by the final classifier, trained with the weaklearner Bubbles or, put another way, the green cloud is the zone where one will find class-1 samples and everything outside that zone belongs to class-2 samples. (The final classifier in this figure is not the one trained with the sample set in Figure 16)

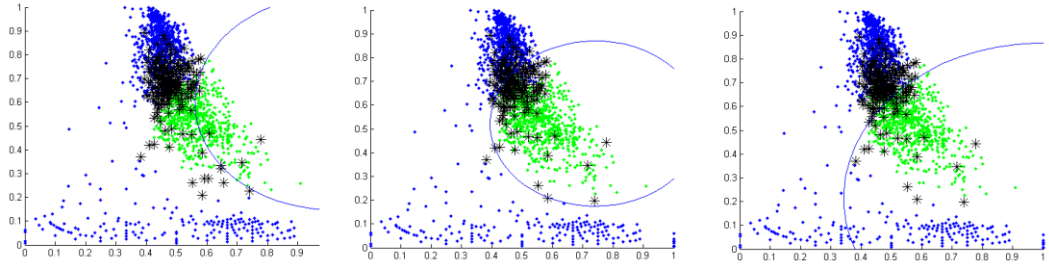


Figure 18 A 2D feature space, the hypothesis in this figures are circles with different radiuses which are taken from three different boosting rounds. The final classifier finds the area which divides the two classes from each other, see Figure 19.

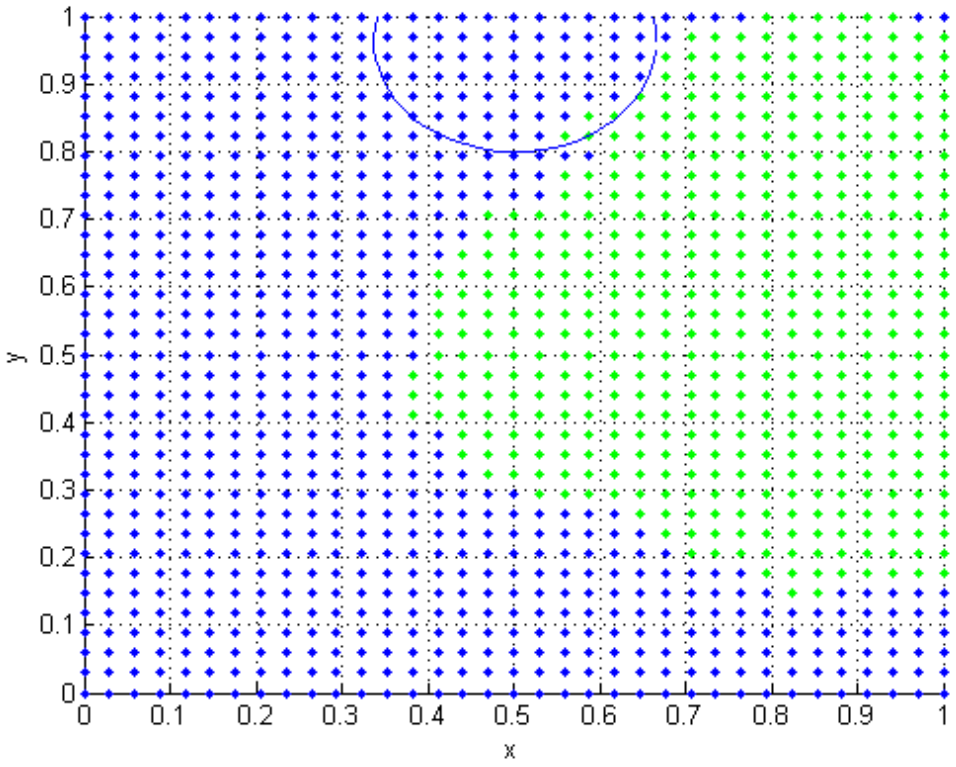


Figure 19 These two areas are the outcome from the final classifier which has been training in 30 boosting rounds on samples seen in Figure 18.

Table 4 Inside the weaklearner Bubbles (3D)

Input: Weight vector w for N labeled samples $S = \langle(\mathbf{v}_1, \mathrm{y}_1) \ldots (\mathbf{v}_\mathrm{N}, \mathrm{y}_\mathrm{N})\rangle$ where $\mathbf{v}_\mathrm{n} \in \mathbb{R}^3$ and $y_i \in \{-1,1\}$. $n = 1, \ldots, N$, *see Table 2: step 1.* Note: The only difference from table 2 is that $\mathbf{v}$ is a three dimensional feature vector.

Set: The number of spheres (points) used, m, the radius step resolution, Δ, the maximal and minimal size of the radius, r_{max} & r_{min}.

The number of different radiuses used for one sphere:

$steps = \frac{r_{max} - r_{min}}{\Delta}$

And the radiuses are stored in:

$radius = \{r_{min}, r_{min} + \Delta, r_{min} + 2\Delta, \ldots, r_{max}\}$

1. Find the volume spanned by $\mathrm{R_x}$, $\mathrm{R_y}$ and $\mathrm{R_z}$, see *eq. (18)*.
2. Generate x,y,z –coordinates for m points that are randomly spread within the volume.
   Return: $\boldsymbol{M} = \{(x_1, y_1, z_1), \ldots, (x_m, y_m, z_m)\}$
3. Generate and Test each spherical shell with the input data. Test if each feature sample is within the sphere:
   For i=1, ..., m
   For j=1, ..., *step*
   For n=1, ..., N
   If $|\boldsymbol{M_i} - \mathbf{v}_\mathrm{n}| \leq radius(j)$
   Then $class_{i,j}(n) = 1$
   Else $class_{i,j}(n) = -1$
   End
   $side_{i,j} = 1$ // this denotes that class 1 ($y = 1$) is within the sphere
4. Calculate the error rate similar to *eq. (17)*, $h_t(x_n) := class_{i,j}(n)$:
   $\varepsilon_{i,j} = \sum_{i=1}^{N} w_n I\big(class_{i,j}(n), y_n\big)$
   , where I is described in eq. (*4*).
5. If the error rate is $\varepsilon_{i,j} > 0.5$ then change the side of the sphere so that class-2 ($y = -1$) is within the sphere:
   IF $\varepsilon_{i,j} > 0.5$
   $side_{i,j} = -1$
   $\varepsilon_{i,j} = 1 - \varepsilon_{i,j}$
   End IF
   End
   End
6. Return a hypothesis, h, with the lowest error value in $\varepsilon_{i,j}$:
   $\mathrm{argmin}_{i,j}\ \varepsilon_{i,j} = \{I, J\}$ //I, J represents the index of the lowest value in $\varepsilon_{i,j}$
   Output: $h = \big\{radius(J),\ \boldsymbol{M}_I, side_{\mathrm{I,J}}, \varepsilon_{\mathrm{I,J}}\big\}$

3.3 Integral image

The Integral image [Crow 1984] was used by [Viola and Jones 2001] as a feature extraction method for face detection. They were inspired by the work of [Papageorgehiou et al. 1998] where they used Haar filters to make a wavelet representation of object classes with wavelet basis functions. In [Viola and Jones 2001] they used integral image to create Haar-like filters that where later used as a features on an image.

If we name a pixel value $P(i,j)$ where (i,j) is the pixel position in an image X then an integral image, that we can call $integral(i,j)$, is created by accumulating all the values in X where the pixels are found to the left and top of pixel position (i,j), the dark area in *Figure 20*, the sum of all pixels in the dark area is stored in $integral(i,j)$.

$$integral(i,j) = \sum_{a=1,b=1}^{\mathrm{i,j}} \mathrm{P(a,b)} \tag{19}$$

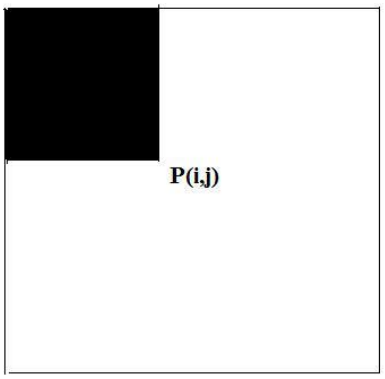


Figure 20 Integral image is storing the sum of all pixel values found in the dark area into $integral(i,j)$.

Then after a complete sweep over all pixel values in X, from $P(1,1)$ to the last pixel in lower right corner, the integral image is completed. With the information stored in *integral* one can extract the sum of any rectangular area within X by only using 4 elements in the *integral* array.

In Figure 21, the sum of an area D in image X is found by only using 4 pixel positions a, b, c and d in *integral*.

$$D = integral(\mathrm{d}) + \mathrm{integral(a)} - (\mathrm{integral(b)} + \mathrm{integral(c)}) \tag{20}$$

The integral image can then be used in a fast way to compare the image areas. One can detect unique image features by subtracting the dark area with the white area, see Figure 22, in various rectangular sizes and combinations. E.g. in [Viola and Jones 2001] they showed that with integral image they could generate 180 000 unique features within a detection window of size 24x24 pixels. The amount of features is far greater than number of pixels available within that detection window.

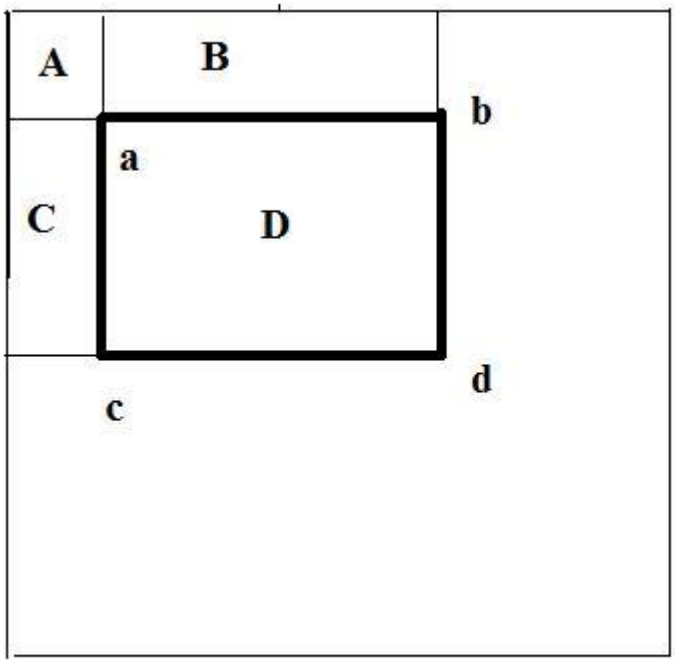


Figure 21 Integral image is used to extract the sum from the area found in D by using 4 reference points a, b, c and d.

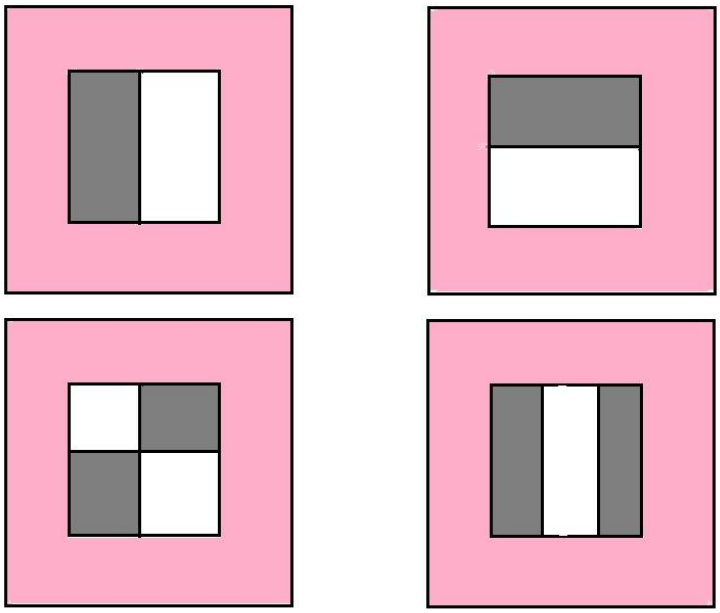


Figure 22 Four detection windows with some examples of how to generate features by comparing areas in the image in different ways. The sum of pixel values in the shaded area is subtracted from the sum of pixel values in the white area.

Table 5 Pseudcode of integral image

Input: *image*= A gray level image of size NxM.

Initialize: A variable *integral* of equal size as input image.

3. Copy the first row and the first column of *image* and store it in the first row and in the first column of a variable *integral*

4. Calculate the accumulating sum of *image* and store the value in *integral* by using the previous stored value in *integral*

For i=2…N

For j=2…M

$sum = integral(i-1,j) + integral(i,j-1) - integral(i-1,j-1) + image(i,j)$

$integral(i,j) = sum$

End

End

Return: *integral*

5. Extracting the sum of a patch area in image use *integral* as in *eq. (20*)

Integral image will be used to find the area that contains lot of pixels that belongs to a defect. All pixels are evaluated by the final classifier one by one. However, one pixel alone is not enough as evidence that a defect exist so an area that contains lots of pixels that belongs to a defect is a more accurate sign.

If the sum of detected defect pixels within the rectangular window exceeds a given threshold then the area will be marked. Most of the extra features that integral image can generate, by combining the rectangular patches as seen in Figure 22, is not used because the defects does not have any typical shape or patterns in the same way as a human face.

3.4 Dynamic programming

Dynamic programming (DP) is an algorithm that can be used to calculate the most cost efficient path between two points. This algorithm can be used in image analysis to track the contour of an object's border. Suppose that a curve is drawn from left to right on an image. The image is represented as a two-dimensional array and the curve's pixel values are low compared to the surrounding pixel values.

Let's say that the DP algorithm starts from the left side of the image. Then it calculates from the image the cost in moving from all the elements in the first column to an element in the second column.

The lowest cost, in moving from an element in the first column to an element in the second column, is stored in a cost accumulating matrix (CAM) and the connection between these two elements is stored in a tracing matrix, see Figure 23.

When DP completes the cost calculation of the second column it continues to calculate the accumulating cost in moving from the second column in CAM to an element in the third column in the image. And this is done recursively until the last column is reached.

The total cost in moving across the image from left to right is stored in the last column in CAM, see Figure 24. The lowest cost in that column should be the end point of the curve because the curve is represented by pixels with low values. The shape of the curve can be found in the tracing matrix which is used to trace back between the columns until the first column is reached [Sonka et al. 2007].

One can add a penalty in moving up or down to the cost calculation so that the traced curve becomes smoother. Without this penalty the DP algorithm might find that the lowest cost is by taking big steps in moving up and down between the columns.

Table 6 Dynamic Programming in image analysis

Given an image represented in a matrix of order MxN. The first column of the CAM matrix is equal to the first column of the image matrix. The cost accumulating matrix is calculated with a weight ***w*** in this example.

For i=2…N

For j=1…M

For y=1…M

$\Delta y = |j - y|$

$costvector(y) = CAM(i-1, y) + Image(i, j) + w\Delta y$

END

$CAM(i, j) = \min_{y} costvector(y)$

$trace(i, j) = y$

END

END

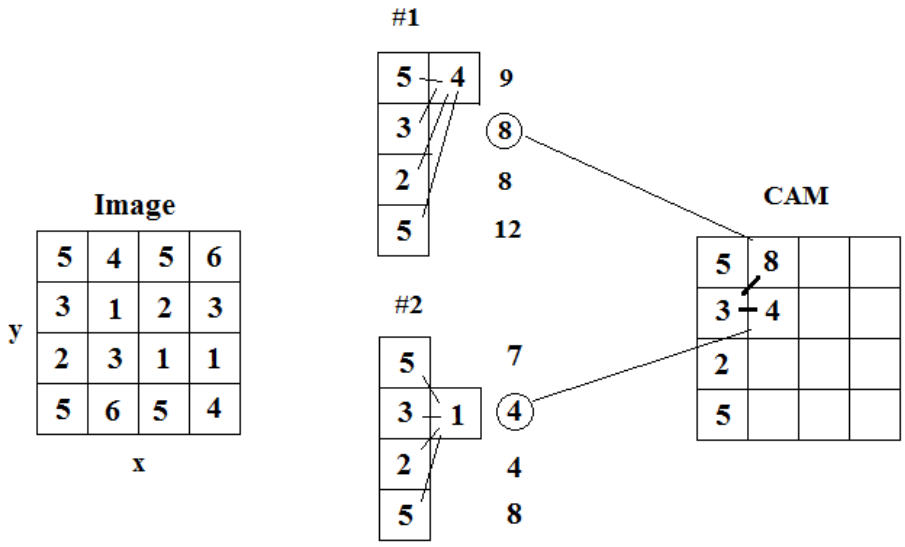


Figure 23 The DP algorithm calculates with w=1 *the cost in moving from all the elements in column 1 (CAM) to the element in column 2 (Image)* step #1, *the lowest cost is stored in column 2 in CAM and the corresponding connection is stored in a tracing matrix (thick line). When column 2, in CAM, is filled the same process is done to fill in the third column by calculating the accumulating cost from column 2, in CAM, to column 3, in image.*

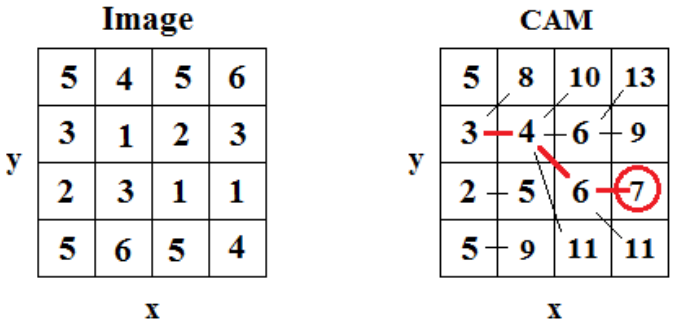


Figure 24 When the CAM matrix is complete the most cost efficient path is found in the lowest value in the last column (red circle) and the path is traced back (red lines).

3.5 The classifying table

One important method that was used to improve the detection speed is to not let the final classifier from *eq. (5)* evaluate each pixel. It would take too much time just to classify each pixel in one image with 10 million pixels. Let's say, if the final classifier is made of thousand hypotheses (thresholds), trained to function with HSV colors, then to classify one pixel we would have to first apply a color model transformation from RGB to HSV and then apply

thousand thresholds. And maybe in addition we would like to use multiple classifiers that work with other color models.

The solution to reduce the computational cost is by directly retrieving the classification for each RGB pixel from a preprocessed table instead of transforming the colors and then classifying them with thousand thresholds.

If each channel in a RGB image is 8-bits then the intensity level of red can be given in any discreet number between 0-255, which is 256 ($=2^8$) ways, the intensity level of green in 256 ways, etc. One can combine all three channels to produce about 16 million ($\approx 256^3$) different colors to choose from. The RGB image was nicknamed "true color image" because of its capability to capture lifelike scenes. A table with 16 million elements was created so that it can represent all colors in a RGB model, with 8-bits in each channel. Every corresponding color in that table was transformed into a color model that a classifier was trained in. The 16 million transformed colors were evaluated by the final classifier *eq. (5)* and the outcome was stored in each corresponding element in that table.

E.g. After completing the training with Adaboost, the final classifier forms a decision boundary or a decision volume that separates the colors from class-1 and class-2. Let say that the final classifier is trained with sample pixels from HSV color space, the input for the final classifier must then be HSV values. The RGB pixel values have to be first transformed into HSV pixel values before using the final classifier on a RGB image.

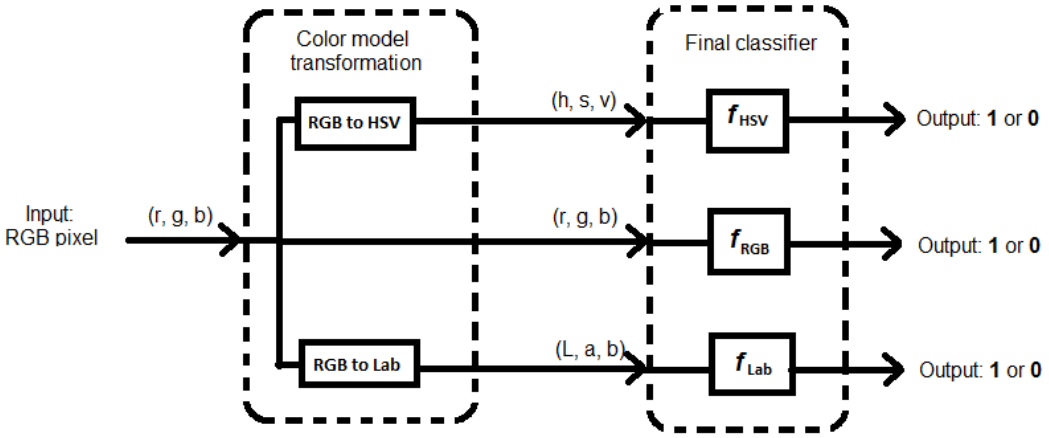


Figure 25 To classify a RGB- image with a classifier that is trained in HSV or Lab color space, one must first transform the input RGB pixel values into respective color model and then forward the values to the classifier to get a binary output.

A classifier table is created to avoid this transformation process which still uses the final classifier that is trained in a HSV color model. The classification

process in the final classifier can also be reduced with a table. The decisions can be recorded and stored in the table instead of letting the final classifier compute and decide each time if a pixel belongs to a certain class. This will reduce the computational cost described above to a process which is a direct link between the RGB pixel value and its respective class.

The classifier table for a final classifier in HSV is created as in Table 7.

Table 7 The classification table

Requirements: a final classifier $\mathbf{H}_{\text{HSV}}$ in HSV color space,
$f(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$,

$f_{HSV}(h, s, v) \rightarrow 1\ or\ 0$.

Color transformations function from RGB to HSV (***RGB2HSV***).

The color transformation table:
Transform all 16 million pixel values from RGB to HSV.
FOR r=0,…,255
  FOR g=0,…,255
    FOR b=0,…,255

$\boldsymbol{RGB2HSV}(\text{r, g, b}) \rightarrow \text{h, s, v}$:color transformation process
$ClassTable(r, g, b) = f_{HSV}(h, s, v)$:creating the classifier table

    END
  END
END

Output: $ClassTable(r, g, b) \rightarrow 1\ or\ 0$

The same method can be applied to a Lab color model or any other color model.

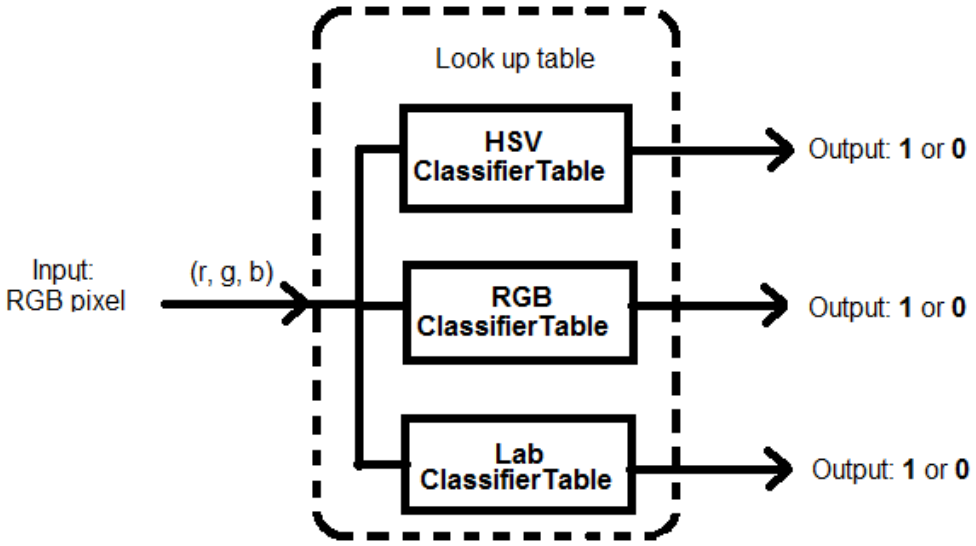


Figure 26 The output from a final classifier is stored in a table so that a RGB pixel can be directly evaluated by looking it up in the classifier table. The information which is calculated in color transformation and final classifier, see Figure 25, is only recorded into respective ClassifierTable.

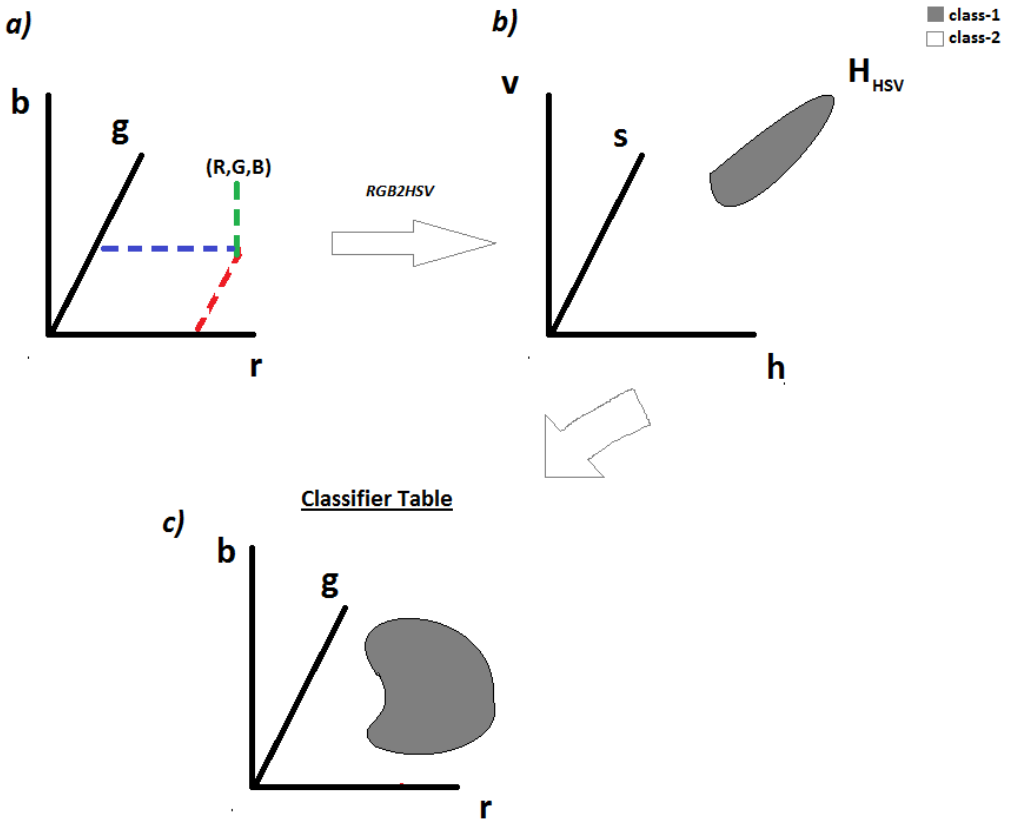


Figure 27 In a), The RGB values extracted from the image must first be transformed to the HSV or Lab color space. The HSV value is retrieved from the transformation function RGB2HSV, b), with Adaboost a final classifier f is trained that shapes a volume (shaded) with the decision boundary in HSV color space so that a HSV value that is outside or inside the volume is assigned to its proper class. In c), the decision boundary created in HSV, b), by f is mapped onto RGB, c). The classifier table c) determines directly what class a RGB pixel value belongs to.

The classification boundary that will eventually confine the colors that belongs to a defect is shaped in the respective color space, Figure 27 b). The decision boundaries can be projected back to RGB , Figure 27 c), so that instead of classifying a RGB pixel by first color transforming it to the respective color space and then use the final classifier one can directly classify it with a look up table.

3.6 SUSAN edge detector

The SUSAN edge detector [Smith and Brady 1997] is well suited for detecting shakes and it is a fast non-linear algorithm that has low noise sensitivity. The edges are found by using a mask that scans the image pixel by pixel. Each pixel within the mask is compared to the masks nucleus (the center pixel). When all pixels within the mask are compared to its nucleus an area on the

mask is pointed out which consist of the pixels that has the same or similar value as the nucleus, this area is known as USAN (Univalue Segment Assimilating Nucleus).

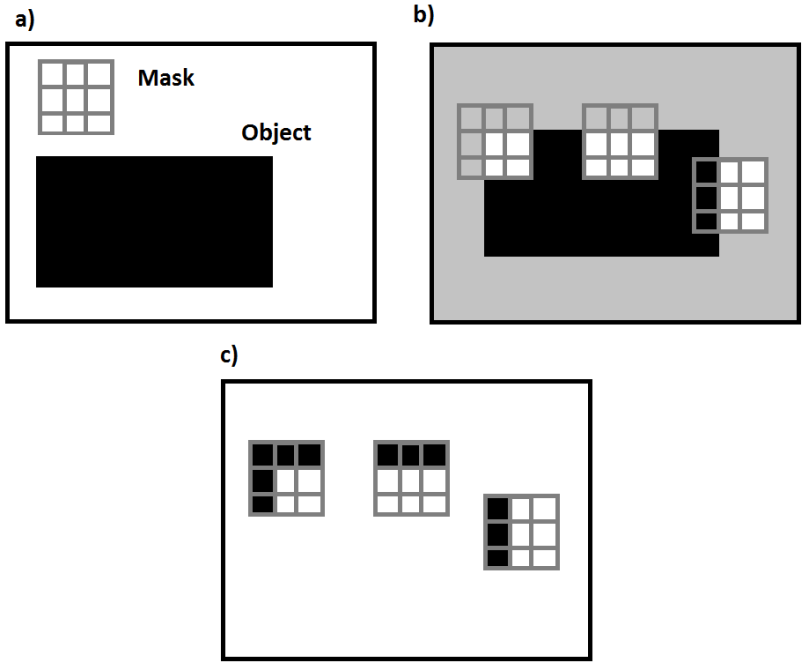


Figure 28 the USAN principle is illustrated with a 3-by-3 pixels mask. b) The mask is tested at three different locations on an image with a rectangular object. If a pixel in the mask has the same value as the nucleus the pixel location will then be a part of the USAN area. c) The USAN area can be seen as the white pixels within the mask.

The edges can be found by looking at the USAN area. When the mask is at a flat region the USAN area will have its maximum size but when it reaches an edge the USAN area will shrink. The edges in an image will be strongly enhanced when the USAN mask is applied to it, with lower values at the edges. The edges are then found by applying a low threshold or by finding the smallest threshold that will have the best edge response.

This method is called SUSAN which is the acronym of "Smallest Univalue Segment Assimilating Nucleus". The nonlinear form makes SUSAN low sensitive to noise compared to other edge detectors because it does not use a derivative or zero-crossing to detect the edges. The edge found in SUSAN is from the USAN area which is a sum of comparison of the pixels in a local region within the mask.

The SUSAN equation is,

$$c(\bar{r},\overline{r_0}) = \begin{cases} 1 \; if \; |I(\bar{r}) - I(\overline{r_0})| \leq t \\ 0 \; if \; |I(\bar{r}) - I(\overline{r_0})| > t \end{cases} \tag{21}$$

Where, $\overline{r_0}$ is the position of the center pixel in the mask and $\bar{r}$ is any other point within the mask on a two dimensional image. $I(\bar{r})$ is the intensity value at pixel position $\bar{r}$ and t is the intensity difference threshold, c is the output. Each pixel in the mask is compared to the nucleus and the sum of all outputs is calculated.

$$n(\overline{r_0}) = \sum_{\bar{r}} c(\bar{r},\overline{r_0}) \tag{22}$$

Where, $n(\overline{r_0})$ is the amount of pixels within the USAN area. The parameter t controls the mask sensitivity to detect edges and noise. The edge is detected when $n(\overline{r_0})$ is lower than a given threshold $T, T \in [1, N-1]$. N is the amount of pixels in the mask.

4 Wood defect analysis & classification

4.1 The big picture of this work

I have described the necessary theories which will be used as a solution in detecting the wood defects. The theories will be combined and implemented in a computer program. The overview of the on-line workflow is seen in Figure 29 and the off-line training of a classifier table is seen in Figure 30.

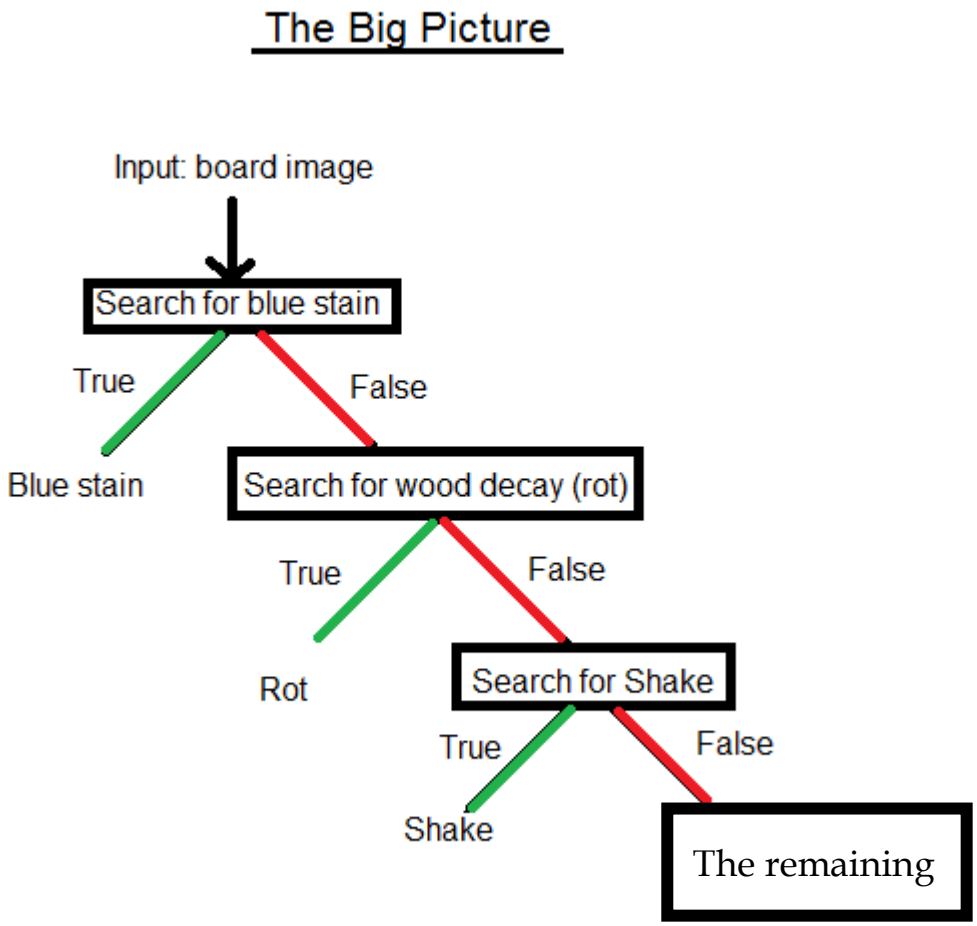


Figure 29 This is the overview of the workflow when the program is on-line.

4.2 Discussion

4.2.1 Large area defects

The rot and blue stain defects will be solved in a different way from shakes. The first step in detecting the rot and blue stain is to separate the colors from everything else. The color information can change with different camera positions because the scenes are not perfectly similar to one another even after a calibration is made nothing is perfectly constant in reality.

The board color tones will vary not only from each wood sample but also along a single board so it is expected that the blue stain and rot color might slightly change in each case. A supervised training on a classifier will identify the defect color. The process of training the classifier table is seen in Figure 30.

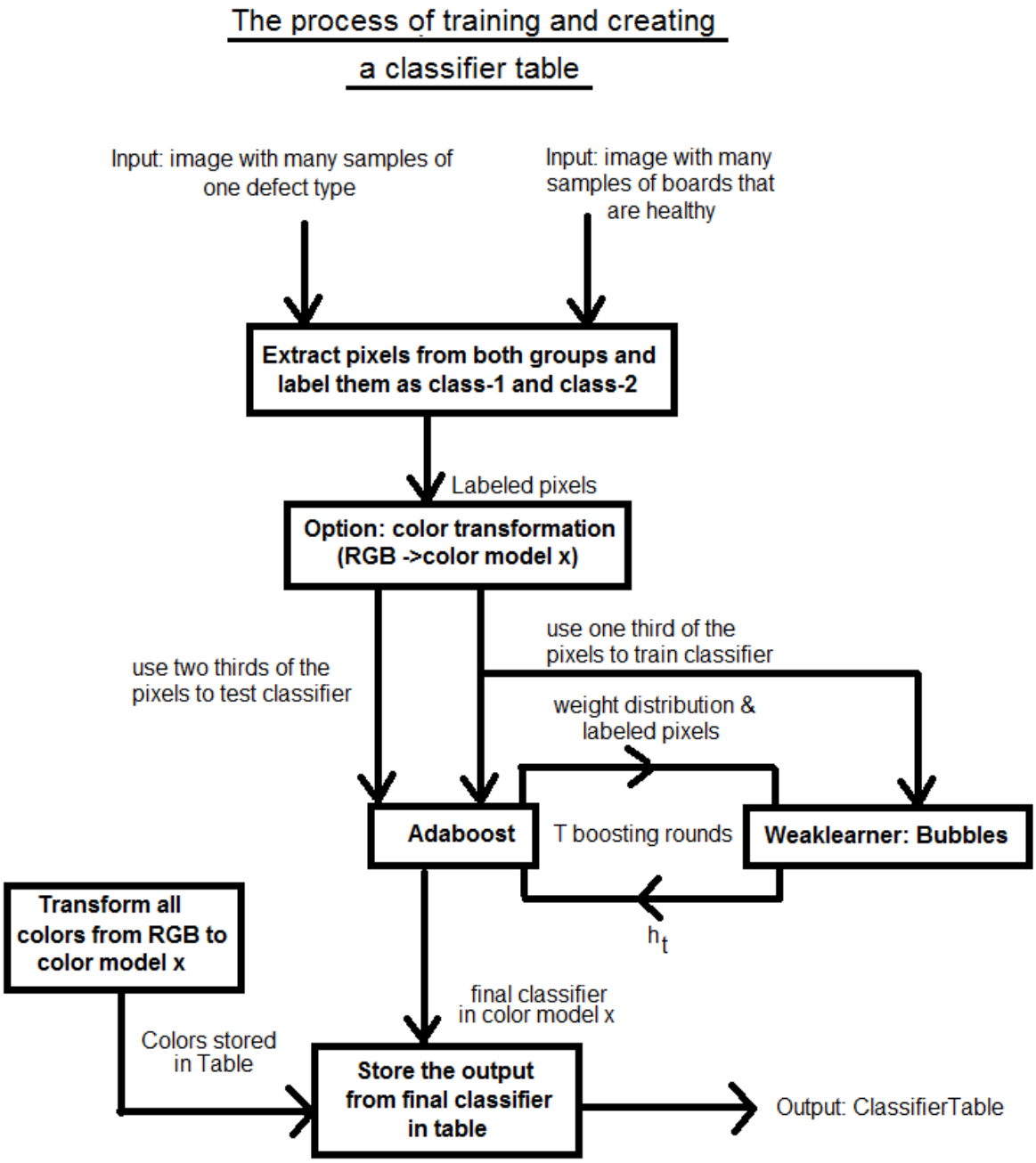


Figure 30 The training of a classifier table (off-line).

4.2.1.1 The search for blue stain

When a classifier table is trained to recognize blue stain colors it can be used in a fast way to classify each pixel from a RGB image. The search is simply looking for the unique color of blue stain on the input image. If many classifier tables are used to classify then each pixel will have a value between 0 and 1, a value close to 1 has a high rate of being a blue stain pixel. Integral image is used to measure the density of pixels within a search window. The complete process is seen in Figure 31.

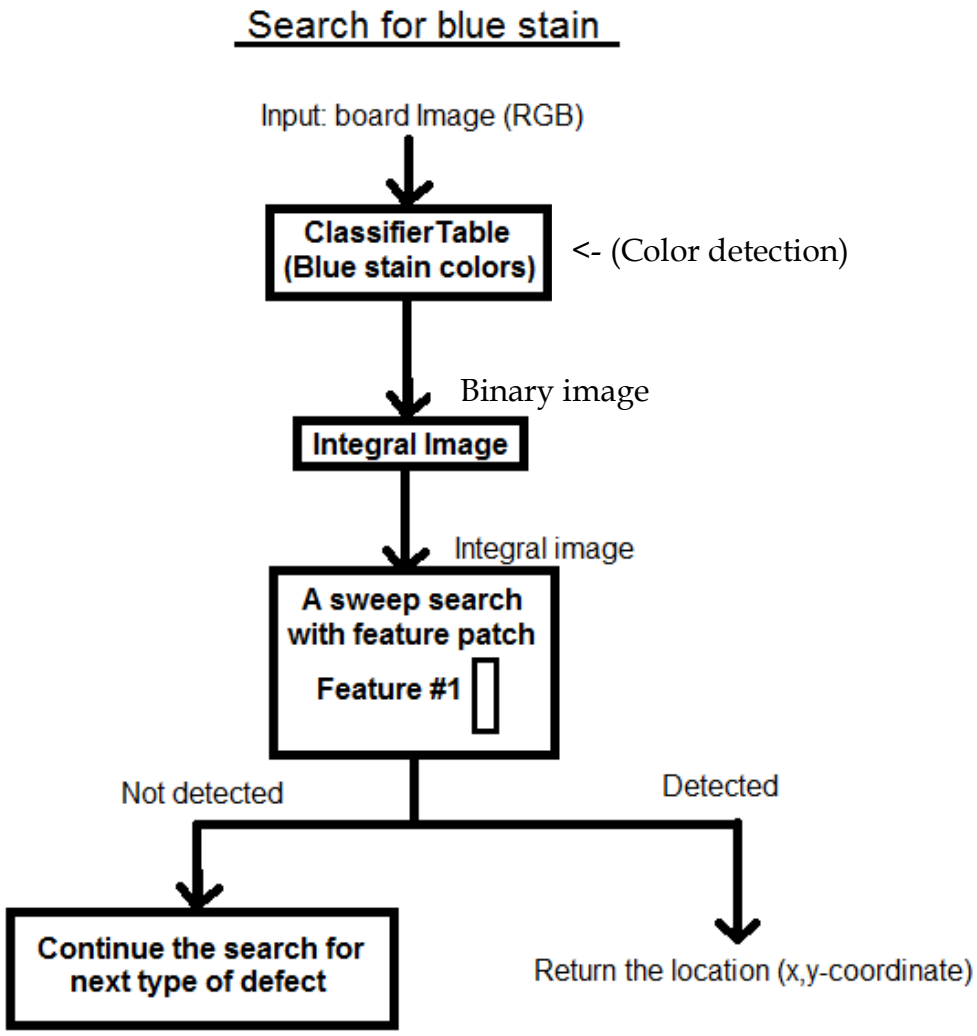


Figure 31 The search process of blue stain is seen in this block scheme.

4.2.1.2 The search for wood decay

This search is similar to the process of searching for blue stain. The difference is that rot has two colors that should be evaluated. The first one is the brown which is the common color. The second color is a key feature which is not always visible, but the search for it is in the same way. This search process requires two classifiers where each classifier is trained to recognize one of the colors. The first part of the process is seen in Figure 32.

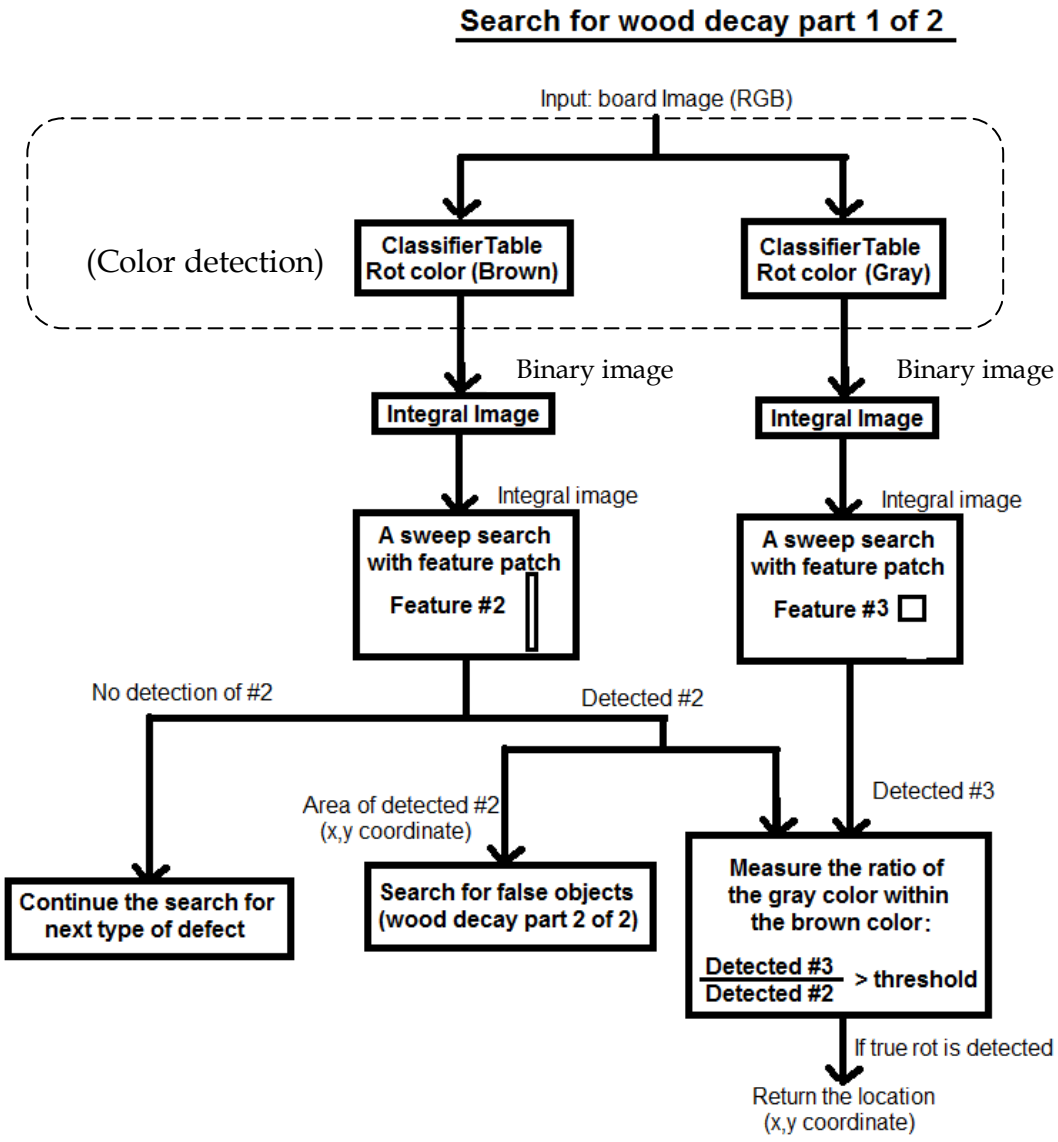


Figure 32 The first part of the search process for rot is seen in this block scheme, the difference from search for blue stain is that this search requires two color detections. The threshold for the ratio of #3/#2 is empirically based, test results shows that the threshold should be higher than approximately 10%, see Table *14*Table *14*Table *14.*

4.2.1.3 A search for false object

The Adaboost algorithm should extract the common color intervals for blue stain and rot so that one can perform a color based segmentation on the image. This could be a satisfying solution for detecting blue stain because of its unique color. But rot is a more complex problem which requires other methods because a normal board can have e.g. annual growth rings that has the same color as rot.

As rot does not have any shape or structure and no pattern, the only option to minimize the false detections is to use methods to actually find the false objects. Two types of false objects “annual growth rings” and “water stains” are detected with the methods integral image, dynamic programming and Hough transform. The search process for these false objects can be seen in Figure 33. A more detailed explanation of how the dynamic programming is used with integral image is found in the discussion of experimental results for detecting false objects.

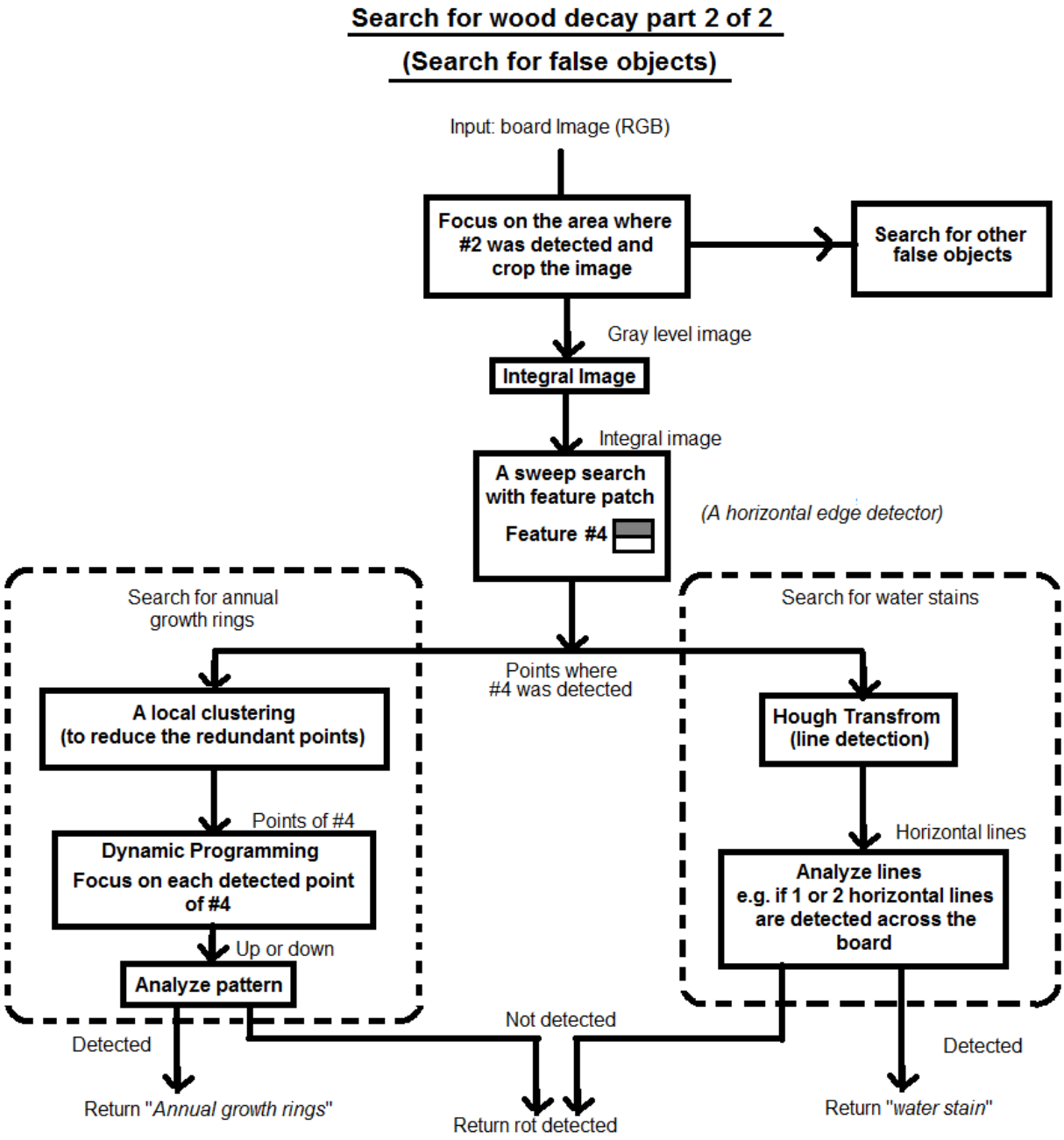


Figure 33 The search process of rot, two types of false objects can be excluded from the search when they are detected with these proposed methods. This will minimize the false detections when rot cannot be evaluated by its colors.

4.2.2 Shake (crack)

Two methods are used to detect cracks. The first one is SUSAN, which is an edge detection algorithm. And the second one is Hough transform, which is a line detection algorithm used to trace the cracks when the edges have been found by SUSAN. The detection can be done later after a further analysis of

the lines e.g. how they are arranged, how many lines are connected to form the crack, how curvy the crack is, etc. One common false object which is detected with the same methods used to detect shake is the pith.

A measure to solve the pith line problem is to search for it indirectly. This can be done by looking at the board cross section. If the pith is there then it will appear as a dark dot. If the dot is close to the edge of the cross section image then it will tell us that the pith should be located on the board surface exactly where the dot is placed. This information will be used to exclude lines that are found where the pith is expected to be located.

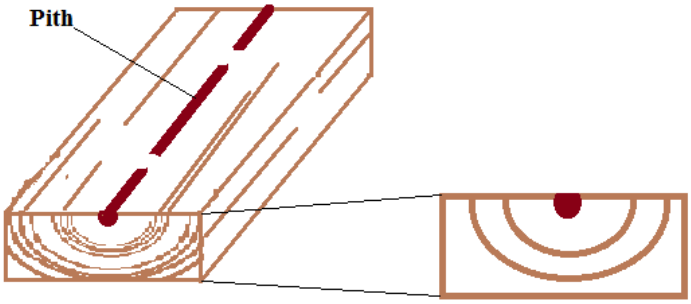


Figure 34 An illustration of the cross section of a board. The dark dot seen in the cross section shows exactly where the pith line is located on the board.

4.2.2.1 Two ways to detect pith

Integral image can be used to detect the pith in the cross section image of a board, see Figure 34. The dark dot can easily be found with the same methods used to detect rot and blue stain. However, the Integral image feature might be different. An additional approach to detect this dark dot is to use an edge detector, let's say that we use SUSAN, and then use a Hough transform for circle detection to find the location of a circle on the cross-section image.

The Hough transform for circle detection can be used to find the center of an arc or a circle if the radius is known. So even if the pith is not visible on the cross-section image one of the annual growth rings can tell where it is located [Sonka et al. 2007]. The search process for shakes can be seen in Figure 35.

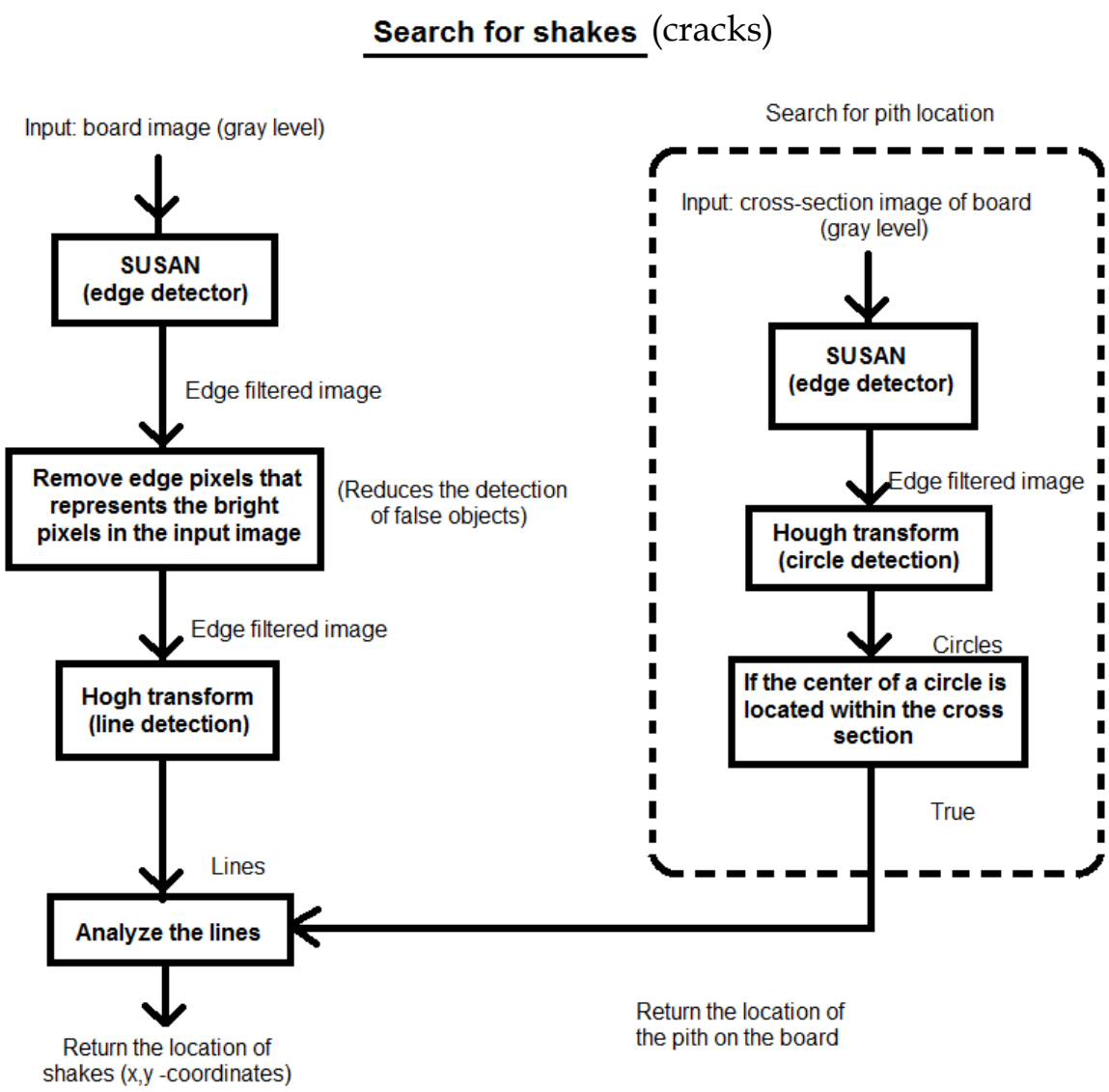


Figure 35 The search process for shakes can be seen in this block scheme. One of the proposed methods to search for the false object "pith" is included.

5 Experimental results

5.1 Training and testing on blue stain

5.1.1 Experimental setup

The colors of blue-stain are found with Adaboost by letting the weaklearner Bubbles train on a set of sample pixels divided into two classes. The sample data is retrieved from images that display normal healthy boards (class-1) and boards that are totally covered with blue stain (class 2). A training set is used, which contain 46 of handpicked blue stain patches and 20 of handpicked normal boards. The two classes where created by cutting out areas that are healthy or stained, see Figure 37.

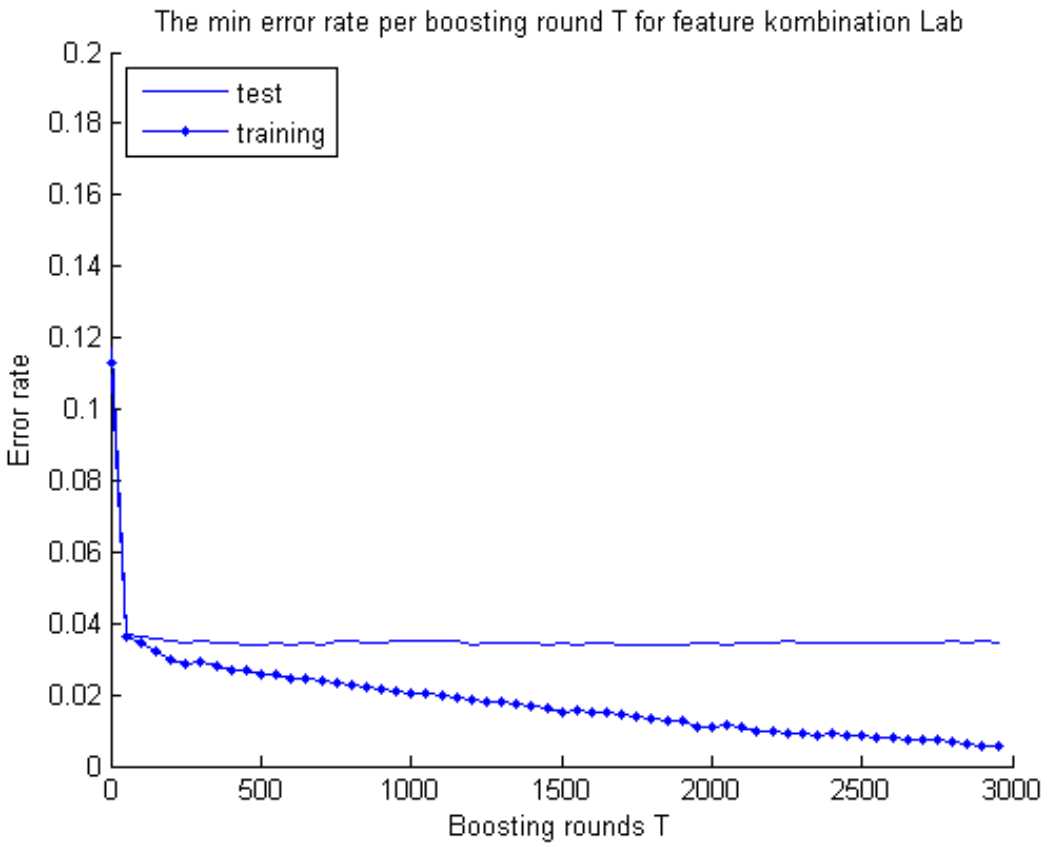
Giving Adaboost the two-class labeled images, it automatically tries to separate the colors of the two classes from each other as good as possible in few boosting rounds.

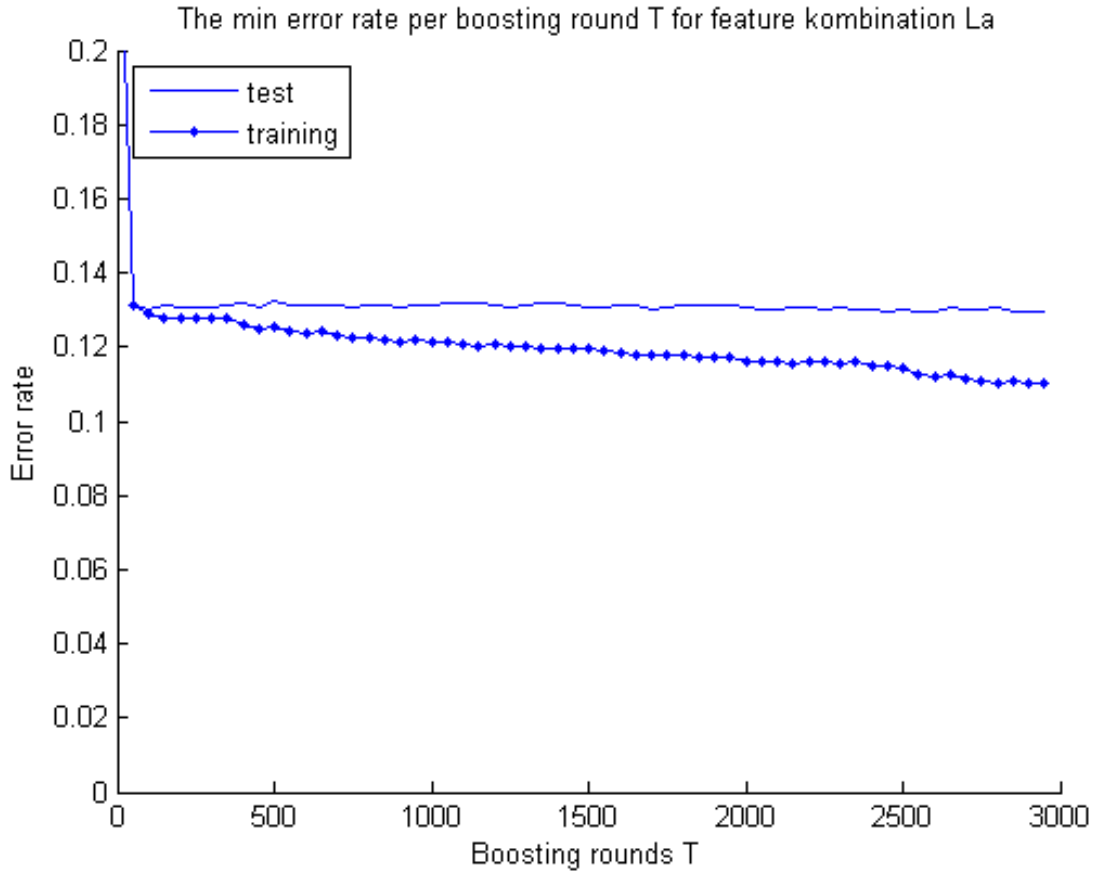
The classifiers are trained with 30 000 samples from the two classes in Lab color model. The Lab color model is used in this experiment just as a demonstration of how well the weaklearner is working in a different color model. Many classifiers trained with different features or color models can be used in *eq. (16)* to build a strong classifier that is based on a variety of features. A cross-validation method is used in the training of a classifier. This will generate many classifiers on the same training data so that the one with the lowest possible error rate can pass the training.

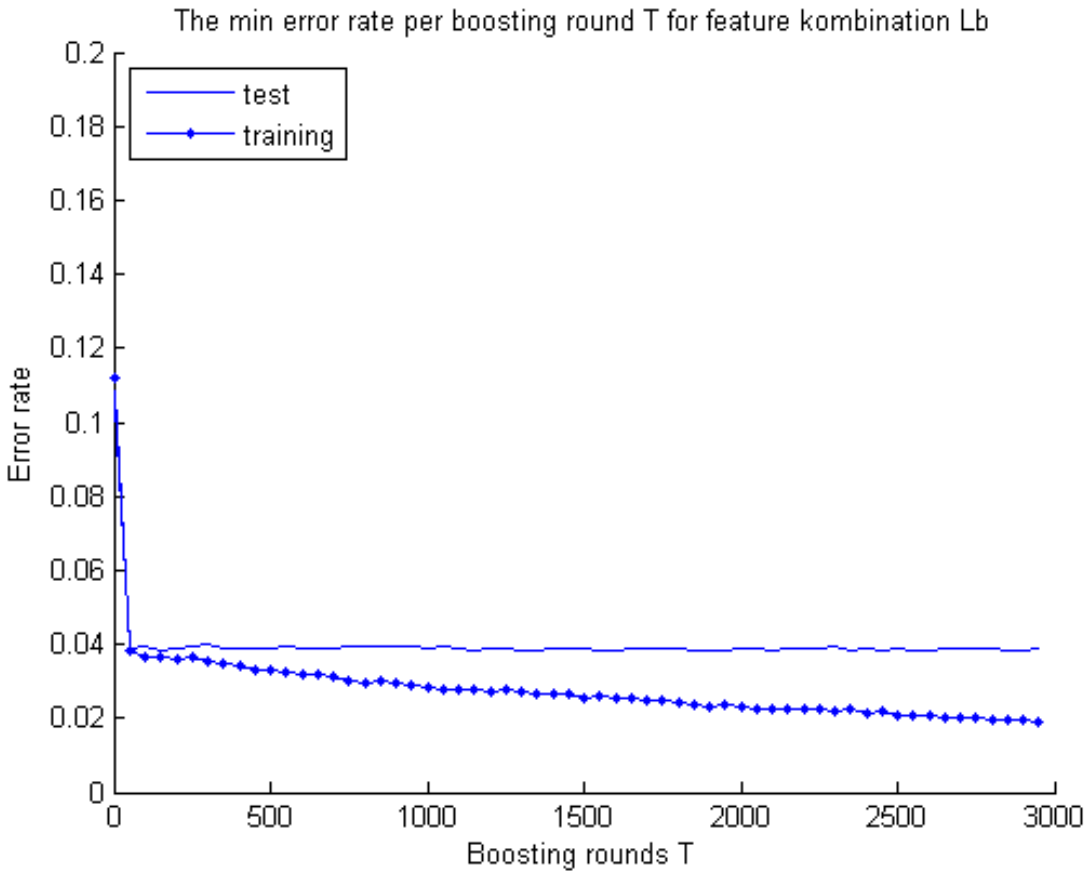
5.1.2 Results

5.1.2.1 Training

The results in Figure 36 shows that Adaboost successively drives down the training error rate while the validation error rate reaches its lowest level at 200 boosting rounds and remains constant throughout the training.


The min error rate per boosting round T for feature kombination Lab
test
training
Error rate
Boosting rounds T
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
0
500
1000
1500
2000
2500
3000


The min error rate per boosting round T for feature kombination La
test
training
Error rate
Boosting rounds T
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
0
500
1000
1500
2000
2500
3000


The min error rate per boosting round T for feature kombination Lb
test
training
Error rate
Boosting rounds T
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
0
500
1000
1500
2000
2500
3000

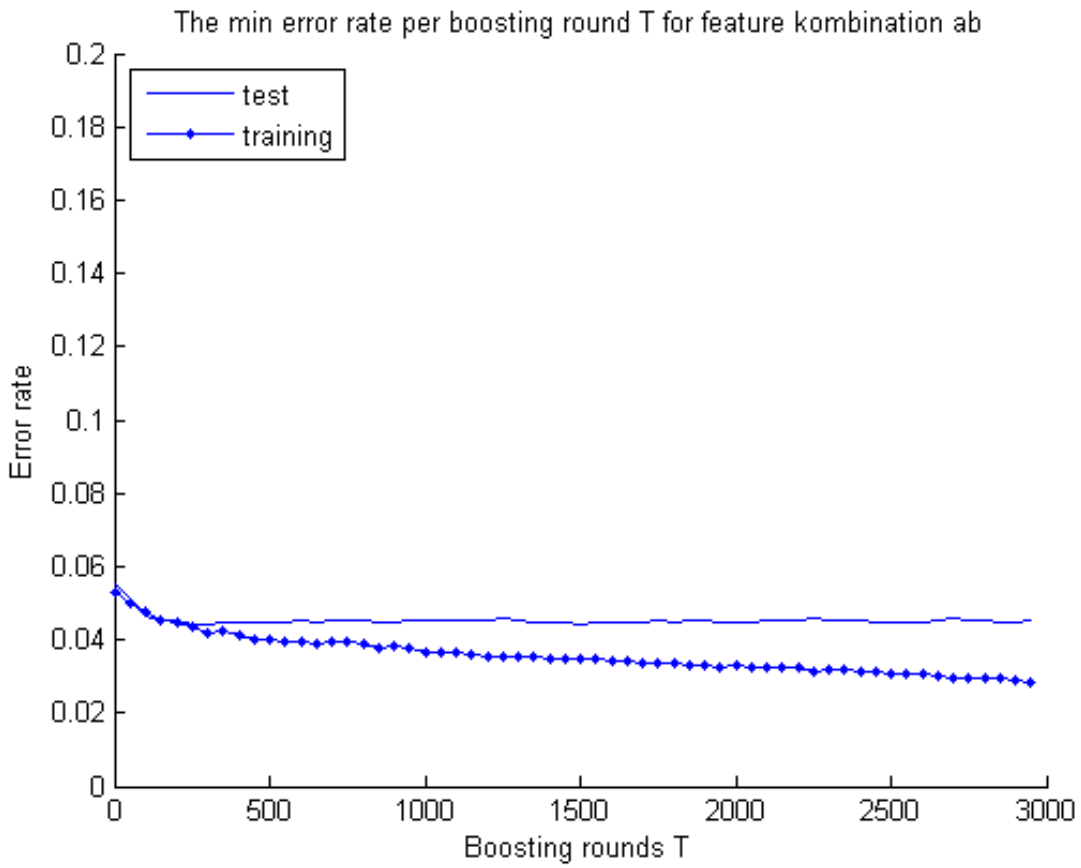


Figure 36 The training results of classifiers Lab, La, Lb and ab. The training error rate is driven down by Adaboost but the validation error rate for each classifier is not improved after 200 boosting rounds.

The two images in Figure 37 are used in the training so that Adaboost defines the colors that belong to blue stain.

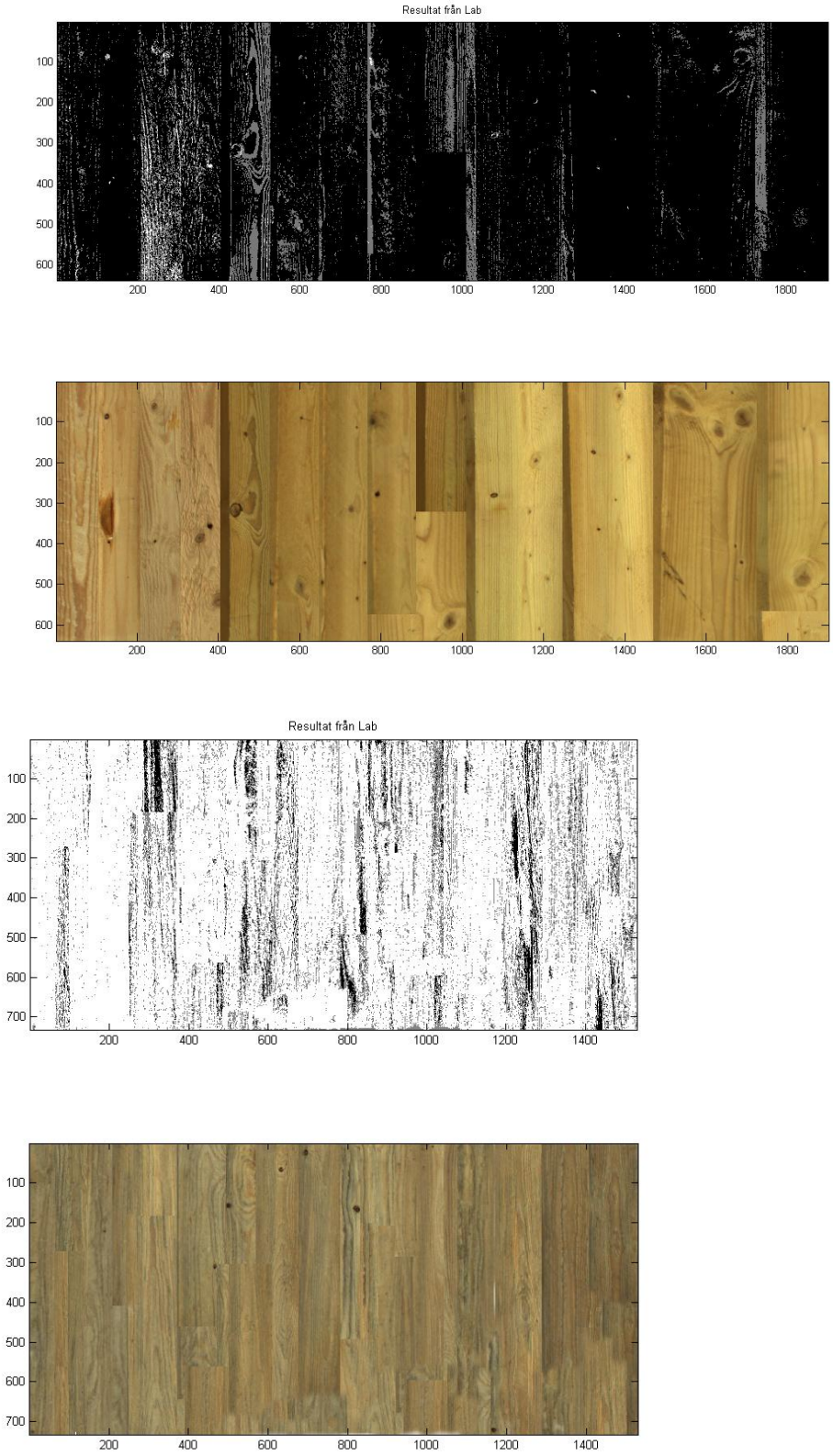


Figure 37 A set of several boards that are normal and healthy (top) and several boards that contain blue stain (bottom) are used as a training set. The colors are first transformed from RGB into Lab color model. Adaboost is then trained in Lab color space. The results shows that a low false detection is made on the healthy boards

training set because Adaboost cannot perfectly separate the two classes from each other, the color samples are overlapping and mixed together to a certain extent. The black and white images represent the output from the classifier. A high intensity (white) means that many classifiers agree on that a pixel value is blue stain color.

5.1.2.2 Testing

When the classifier table is done it will classify pixels directly from a true color image. The output will be a black and white image with pixel intensity range of [0, 1] where 1 is white. A high intensity means that many classifiers agree on that the RGB pixel value is associated with a defect's color.

The intensity in the black and white image represents a voting from all classifiers. A classifier that has a low validation error rate in the training has a higher influence in the voting according to *eq. (*16*)*.

One pixel alone could not verify that blue stain exist on the board so a local area in the black and white image is examined with Integral image. Blue stain patches should fit into a rectangular window with its length and height predefined. The rectangular window will work as a detection window that sweeps over the black and white image. It is used to calculate the sum of all pixel values within the rectangular area. If the sum exceeds a certain threshold then the area that the rectangular window covers should be marked as an area that contains blue stain. Integral image is a well suited method to detect patches because first of all it is extremely fast, one can easily change the detection sensitivity and by changing the length and width of the rectangular window it is possible to filter out objects that should not be detected as blue stain. An acceptable window size is approximately around 20x200 pixels large. This window size is used in the simulation.

Figure 38 The true color image of a board with a faint appearance of blue stain (bottom) is classified by the classifying table. One can see that the pixels that belong to blue stain in bottom image are detected and represented with high intensity in the

black and white image (middle). Integral image with its rectangular window sweeps over the middle image and detects the areas that have a high concentration of white-gray pixels and marks it out with a red color, see top image. The locations of the red objects are found within the green frames in middle image which represents the location of blue stain. The classifiers high sensitivity for blue stain colors makes the classifier able to detect objects with similar colors e.g. tree knots and edges. The detection of tree knots is easily avoided by choosing a long and slim feature window in Integral image, but the edges of the board are also long and slim so they fit exactly into the window. In this image a small part of the edge is visible and it is detected (small green detection window to the right).

Figure 39 An example from the detection program.

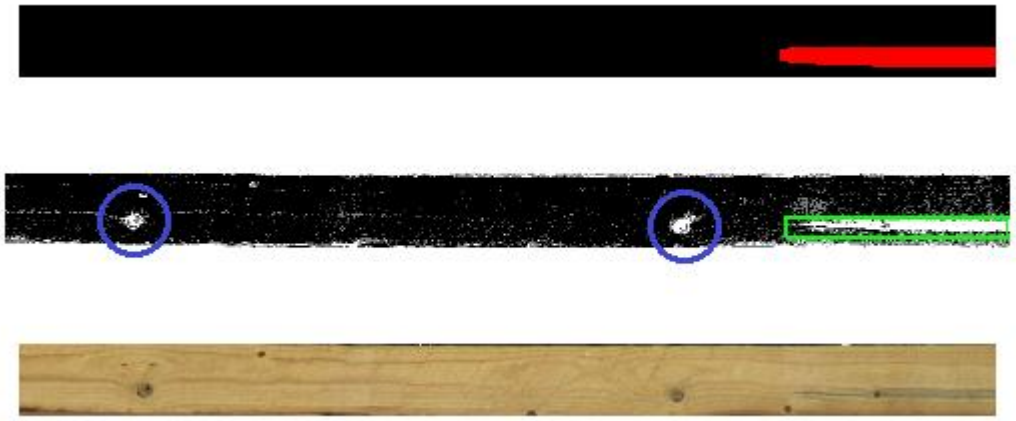


Figure 40 Sometimes tree knots (blue circles) has the same color as blue stain according to the classifiers, to avoid a detection (green) of these knots a thin rectangular window is used in the scanning process. The small round objects fit into the long and slim window but the sum of the white pixels in that window is to low so they are always neglected which can be seen in the top image.

5.1.3 Discussion

The detection is only working on the board surface and cannot be used on images that include the area outside the board because the classifier is trained to recognize the difference between a healthy board and blue stains. It's possible to include images of boards and their surrounding in the training but it's preferred to only use the board surface. The sample pixels from the background might have the same color as blue stain and Adaboost would have to separate these colors from each other and that is not advised.

Two classes with very close sample values will train the classifiers to be very sensitive and would be impractical to use. A change in luminosity can disturb the classifiers and they would not be valid because the training with Adaboost is done exclusively for the scene where the light is evenly distributed. The classifiers could of course be trained on scenes with different light conditions but a large amount of the sample pixels from the two classes might be equal to each other.

The edge of the board has a sharp 90 degree angle and the light that is reflected from the edge surface is weak compared to the rest of the board. The classifier sees these dark edges as blue stain because the colors are similar so the classifier should only be used on the surface that is faced towards the camera. Figure 38 shows a nice example of this effect. The edge is included in this image, which is seen in the smaller frame to the right (green). This frame indicates a detection of blue stain but it is actually a part of the edge, which has a lower brightness compared to the rest of the board surface.

5.2 Training and testing wood decay

5.2.1 Experimental setup

The initial step in detecting rot is similar to the method used in detecting blue stain. Adaboost is training the classifiers on a set containing two labeled images.

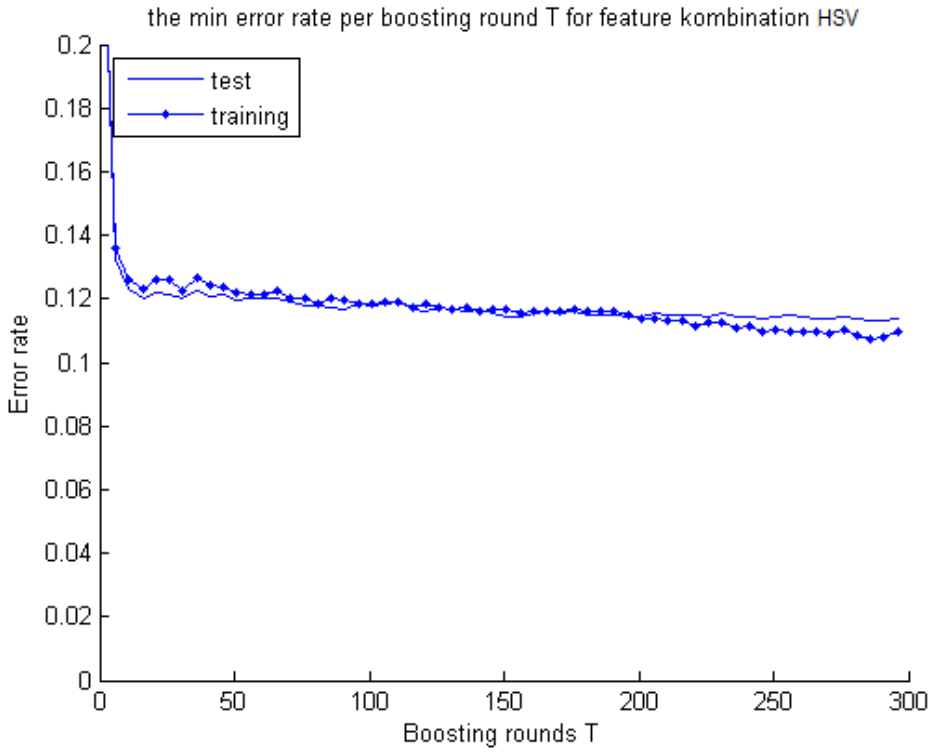
The first image contains samples of decayed wood and the second image contains samples of boards that have everything else that is not rot, except water stains and large brown patches because they can mislead the classifier. This step is mainly used to recognize the wood decay colors and has nothing to do with the shape.

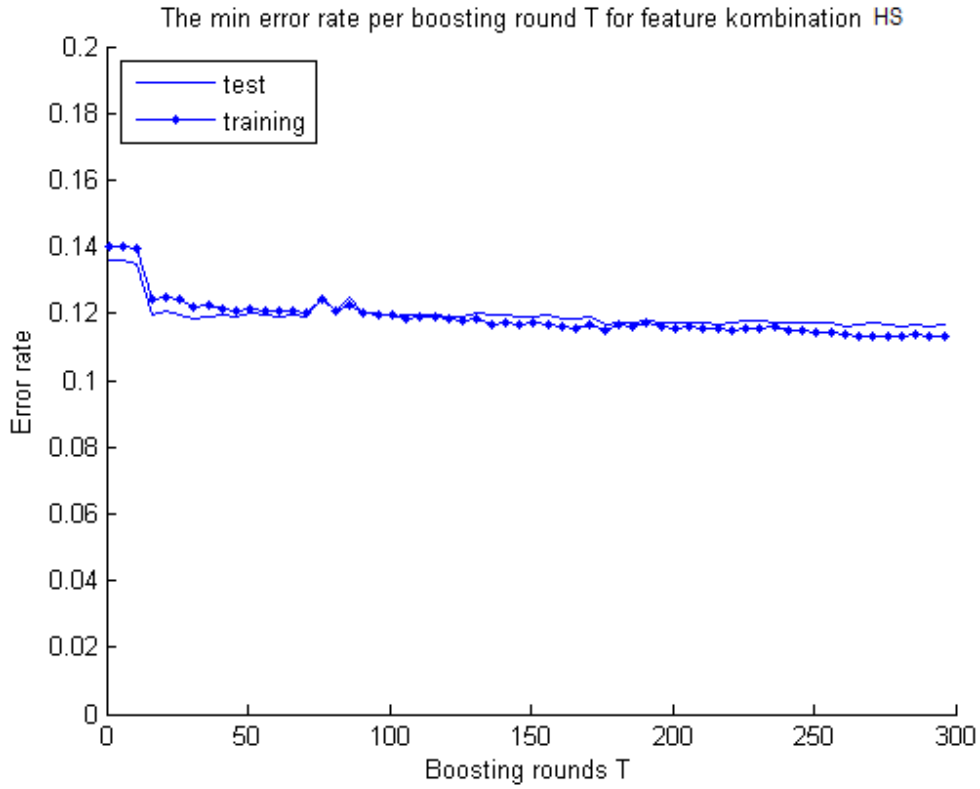
The amount of pixel samples of healthy boards is around five times more than the pixels representing brown rot. This is to deliberately minimize the detection of false objects (the healthy color). The classifiers decision boundary is forced to move away from the healthy pixels. The effect of this uneven distribution of class samples can be compared with a training of equal amount of samples from both classes, see Figure 42.

5.2.2 Results

5.2.2.1 Training

The training results from Adaboost can be seen in Figure 41 with equal amount of samples in both classes.


the min error rate per boosting round T for feature kombination HSV
test
training
Error rate
Boosting rounds T
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
0
50
100
150
200
250
300


The min error rate per boosting round T for feature kombination HS
test
training
Error rate
Boosting rounds T
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
0
50
100
150
200
250
300

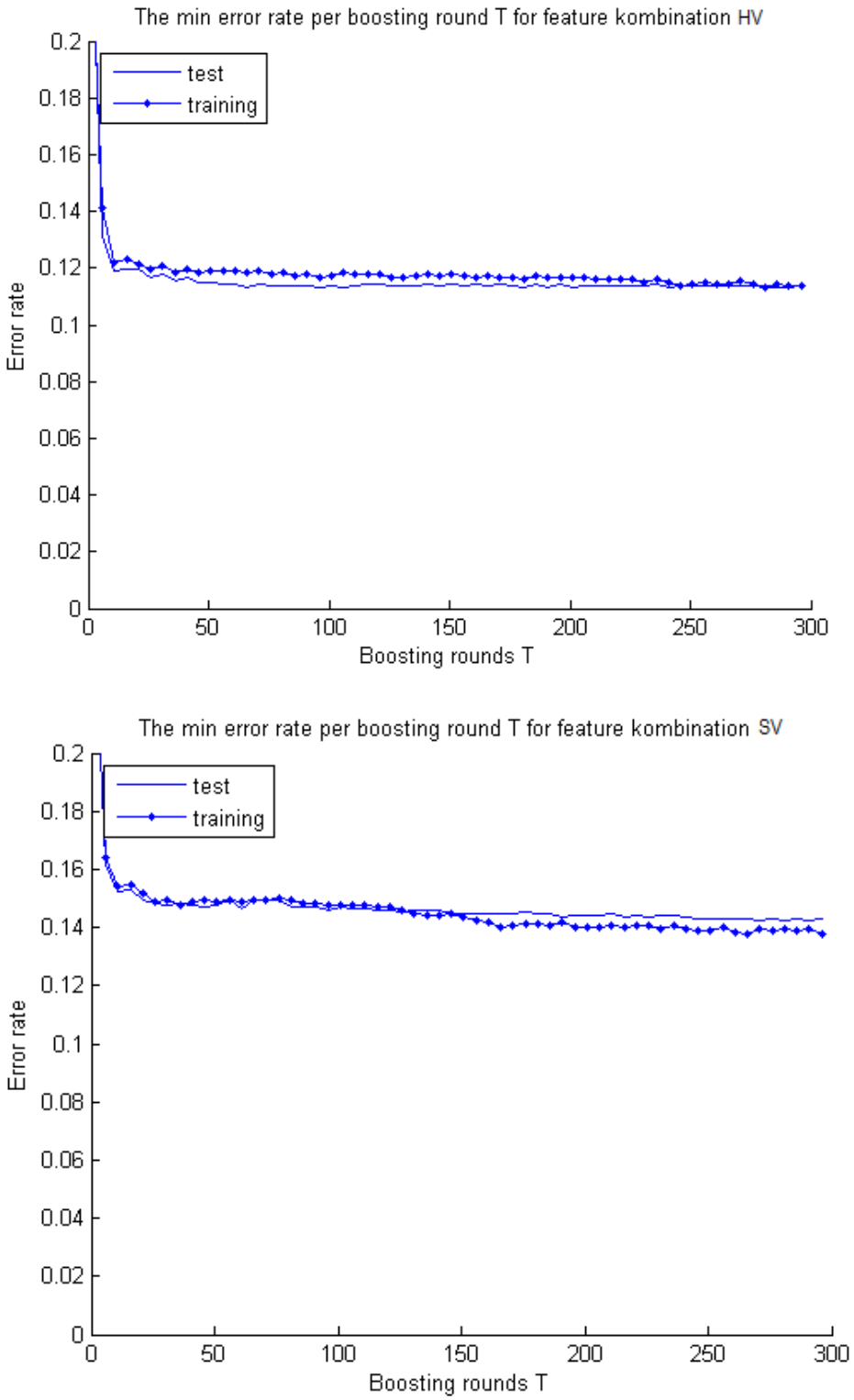


Figure 41 The training results of classifiers HSV, HS, HV and SV on rot with equal amount of pixel samples from both classes. The final training error converges at a higher error rate compared to the blue stain results. This is because of the colors of rot, which is similar to many other objects that can be found on healthy boards.

The high training error rate is caused by a high rate of mixture between the classes. The pixel values in both classes are close to each other so some of

them are overlapping and mixed together. Two trainings were carried out, one included an equal amount of samples in both classes and the other one had five times more of pixels from the healthy boards. The same detection window is used in all images with the same sensitivity and size. The results can be seen and compared to each other in Figure 42.

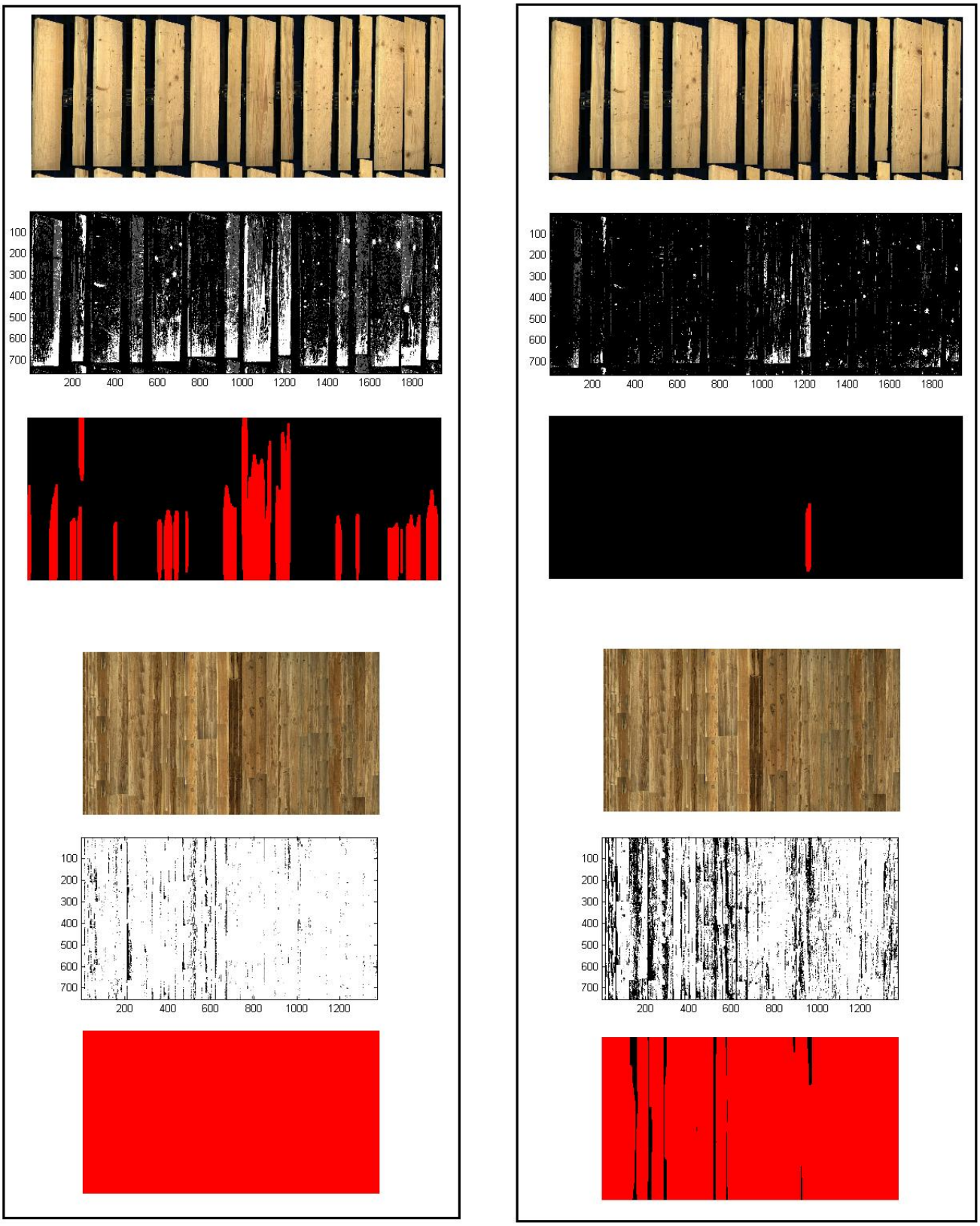


Figure 42 The results from the two trainings can be compared with each other. The left image is the outcome from the training with an equal amount of samples. And the image to the right shows the outcome from the training with five times more of the

samples belonging to the healthy class than the rot class. The gray level image shows the voting from the classifiers in HSV color model. The red color represents the detected area when a 10x300 size detection window is used.

When rot is well established a small proportion of it can appear with an extra color that is gray-like. This extra color can be exploited in the detection and is an important factor because rot does not have any other visible feature. This extra color is a key hint when searching for rot.

Figure 43 This well established rot shows extra gray-like color, which can be seen within the white marking.

An extra set of classifiers are trained to detect the gray-like color with Adaboost. The difficulty is to extract the gray-like pixel values without including any brown-rot colors or healthy-board colors. When a sample image is created it is used to train the extra classifiers in the same way as in detecting the blue stain colors.

5.2.2.2 Testing

By using these extra color classifiers combined together with the rot color classifiers one can assume with a high probability that a detection of a large area with brown color that includes some fraction of gray-like color, more than 7-10 % is decayed wood.

The threshold value 7-10% is empirically based, see *Table* 14.

The second stage in the detection is to use Integral image in the same way as described in section *5.1, results for blue stain*. When it comes to rot the false objects can be very large almost as large or small as the shape of rot. The first detection window size is 10x300 pixels large to avoid most of the false objects that are brown. The detection of the gray color has a window size of 10x30 pixels so it can even detect the small patches.

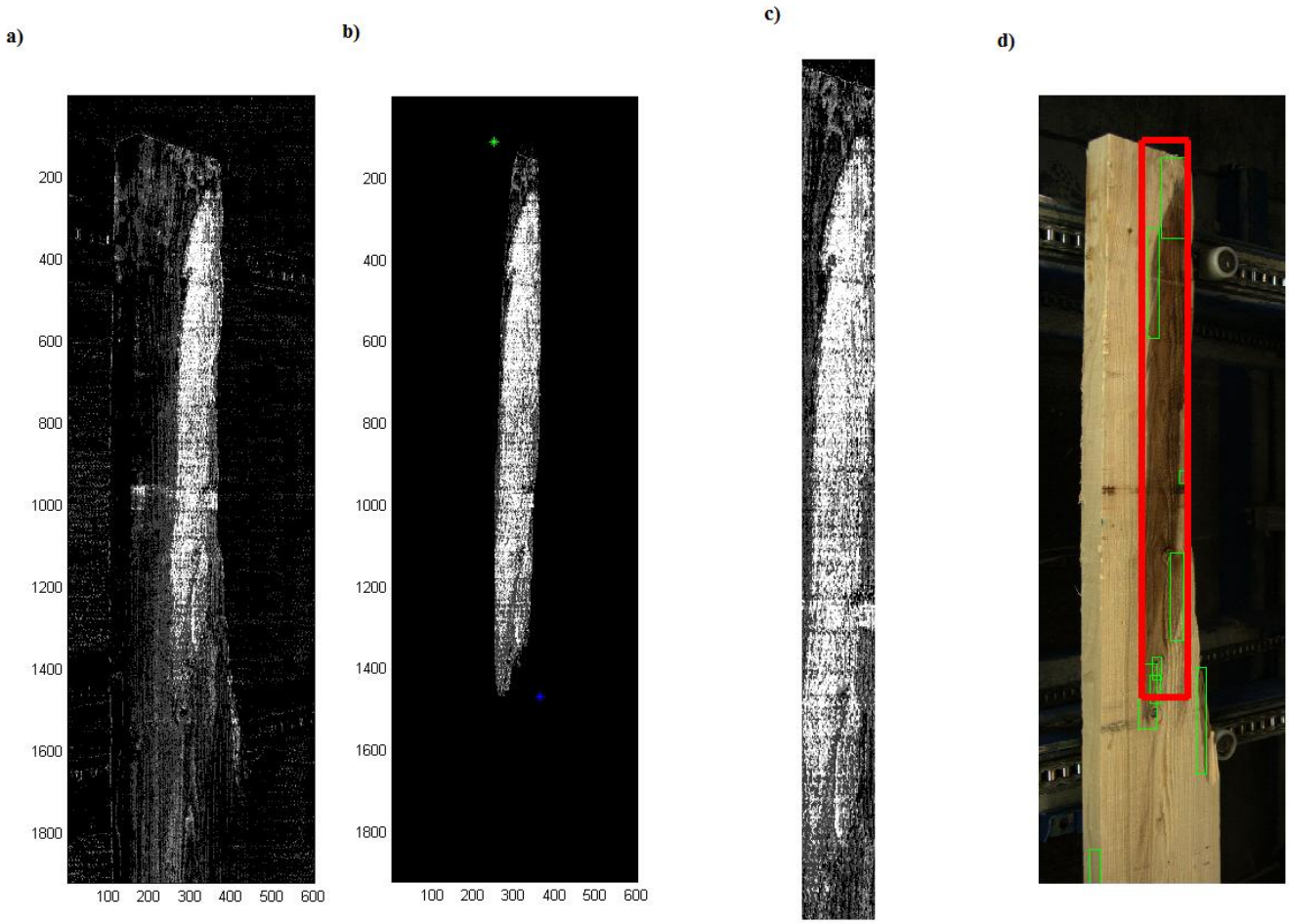


Figure 44 The original RGB image, d), of a board contains rot. The outcome from the classifier table (step 1) can be seen in a) and the area detected by integral image (step 2) is in b). The area detected in step 2 is within the red window in image d). The detection of the gray-like color is found within the small green windows in d).

The next thing is to analyze the area which has the color of rot according to the classifiers. Rot has no special shape feature so an exclusion method is used to keep out the false objects as good as possible from the detection. If there is a sufficient amount of training images of rot then it is possible to use Adaboost again to train a new type of classifiers that can adapt to the shape features (if there is any). Some basic features that can be useful with the methods used so far are listed in *Table 8*.

Table 8 basic feature measurements

Feature	Description
The total intensity, In the black and white image, within the area detected as rot (red frame in *Figure 44*).	Sum(intensities).

Detected area.	Area.
The height of the detection window (red frame in *Figure 44*).	Height.
The width of the detection window (red frame in *Figure 44*).	Width.
The total number of detections of brown rot color with Integral image within detected area (red frame in *Figure 44*).	Number of brown detections/detected area.
The total number of detections of the gray-like color with Integral image within detected area (red frame in *Figure 44*) window area.	Number of gray detections/detected area.

These features can be combined together to find out if there is a difference in the shape of rot and false objects as e.g. water stains. More of the test result of rot can be seen in APENDIX.

5.3 False objects associated with wood decay

There are two types of objects that can cause false detections that are very common on boards. These objects are the annual growth rings and water stains. The detection of these objects is made by two different image analysis methods.

5.3.1 Annual growth rings

The method that was used to detect the growth rings is dynamic programming combined with integral image. The DP algorithm is used to detect the direction of the wave pattern if it is upward or downward on the image by tracing the borders of the growth ring. The DP algorithm starts from one side of the image and calculates the cost until it reaches the opposite side and then traces the curve back to its start location. It is expected that the direction of the curve is known before using DP. What I mean is, if it goes from north to south or from west to east. Only one curve should be present in DP's search window or else DP might jump between two curves and fail to trace one of them. The interest lies in extracting the pattern of annual growth rings, if they are pointed upwards or downwards.

The search area on the image was reduced to minimize the heavy computation that was carried out by dynamic programming. When step 2 is reached, the detection window is focused on a brown patch. Let's say that this

brown patch is not rot and only the pattern of some usual annual growth rings.

The extreama of each curve, from the growth ring, is detected by using Integral image. This method is used once again but with a different detection feature, which can be seen in Figure 45 (horizontal edge detection). This is to minimize DP's search area and also to let DP know where on the detected area it should start tracing the border of the annual growth ring, if they exist.

This integral image feature measure the intensity within the two patches and checks if there is large difference between them. If the difference in brightness between these two areas is large enough then we will know if it is a horizontal border, see Figure 47.

The number of detected edges, with this integral image feature, will be many and some of the detections will be located close to each other. One can use a simple clustering method that finds an average local point to avoid letting DP process the same area multiple times.

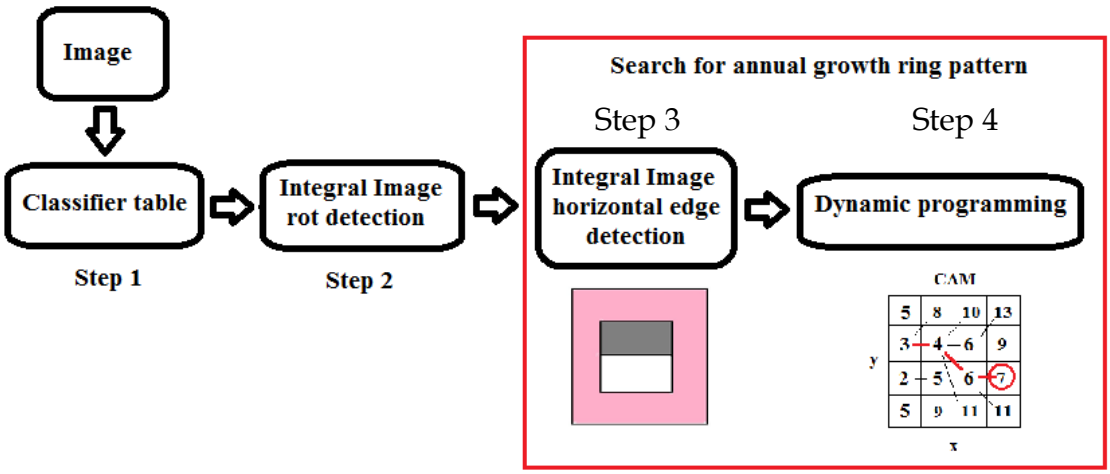


Figure 45 The search flow of annual growth ring. The horizontal edge detection is used to locate horizontal edges on the waves by comparing the intensities within the two areas (shaded and white). This is to minimize the search area and specify exactly where on the image DP should detect the pattern.

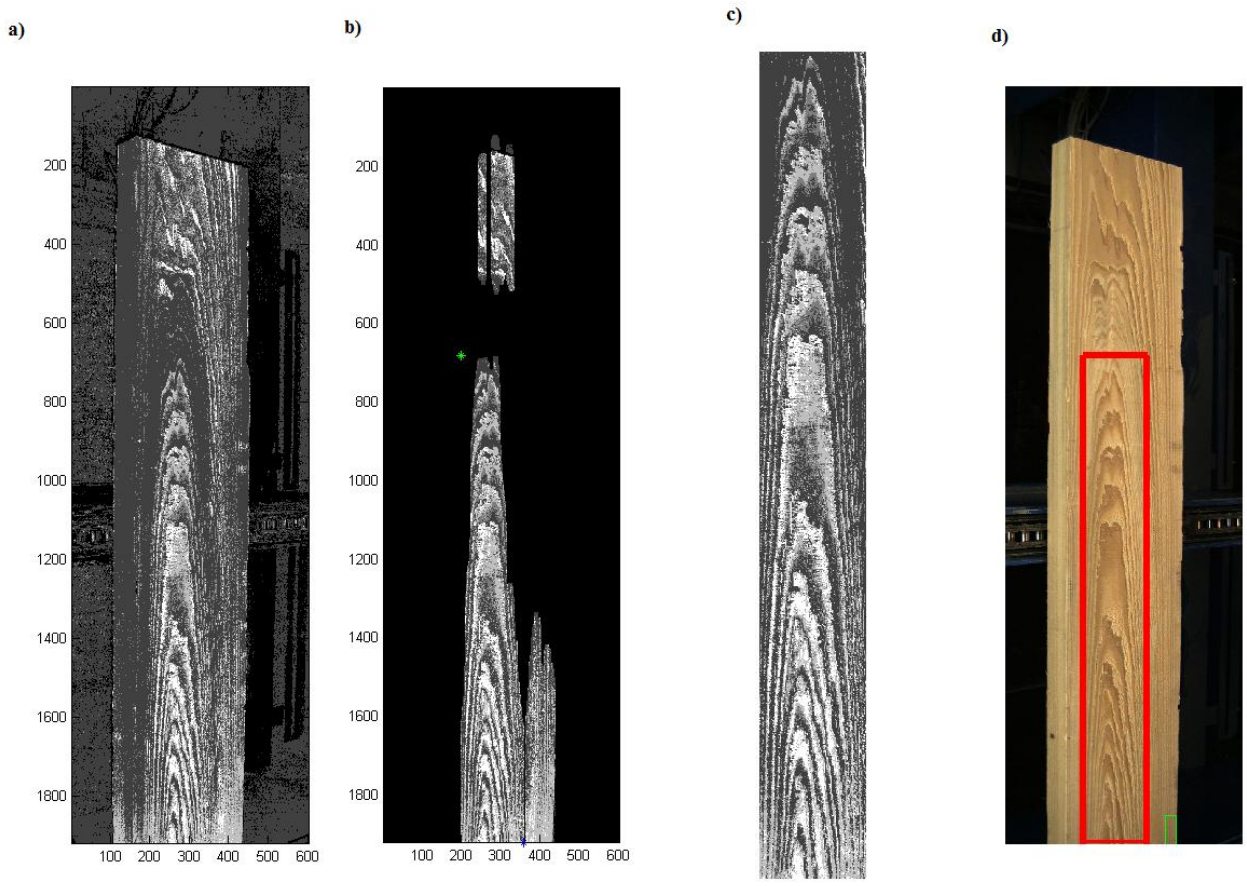


Figure 46 The RGB image d) on a stained board is processed by step 1 and step 2. The outcome from the classifier tables can be seen in a) (step 1) and Integral image is seen in b) (step2). The first area to be found by step 2 is in c), which is the area within the red frame in d). The area contains pixels with the same color as rot according to the color classifier.

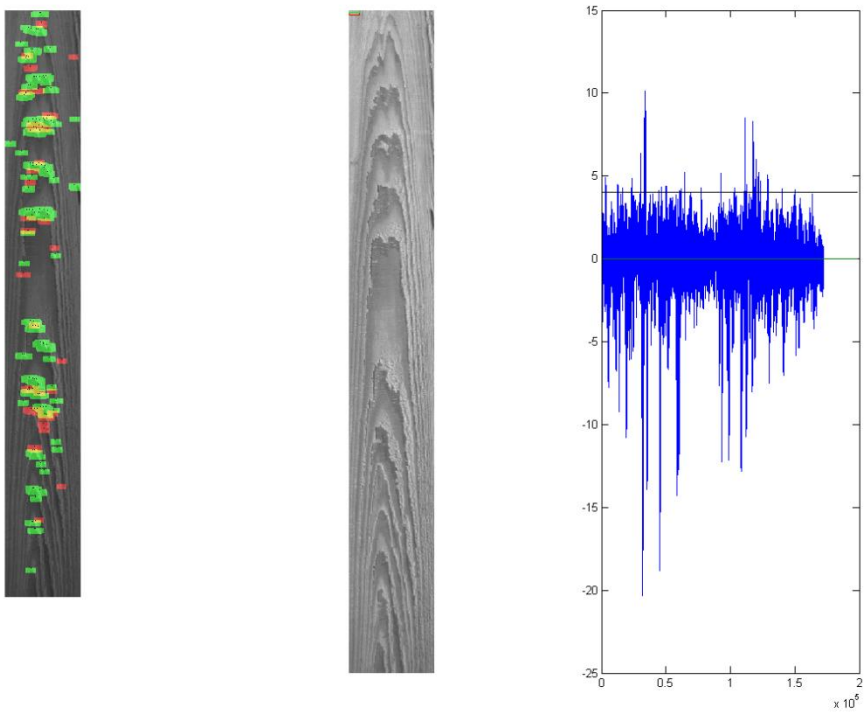


Figure 47 The horizontal edge detection (step 3). The image in the middle shows the annual growth ring pattern. The size of the feature used in integral image can be seen in the top left corner. In this case it is only 5x20 pixels large. The left image shows the detections made by Integral image. The red or green color indicates if the area that is scanned goes from dark to light or vice versa. The image to the right shows the sensitivity threshold for the brightness difference within the feature window.

5.3.1.1 Experimental setup

The window size of 50x50 pixels will be used in DP. This window will focus on the detected area by Integral image, see Figure 47. The size of the 2D-array, 50x50, requires heavy computation by DP so it is minimized by resizing the image part which it is focused on. This will increase the detection speed and the outcome of using DP will almost be the same, if the size is not too small so that nothing will be recognized.

The cropped image is resized to 25% and then split in half vertically because only one curve should exist when DP processes the array. Each half is then given to DP and the border lines that are detected will be compared to each other. The direction of the lines toward each other, outward or inward, will tell if there is a wave like pattern that has an upward or downward direction in the image, see Figure 48.

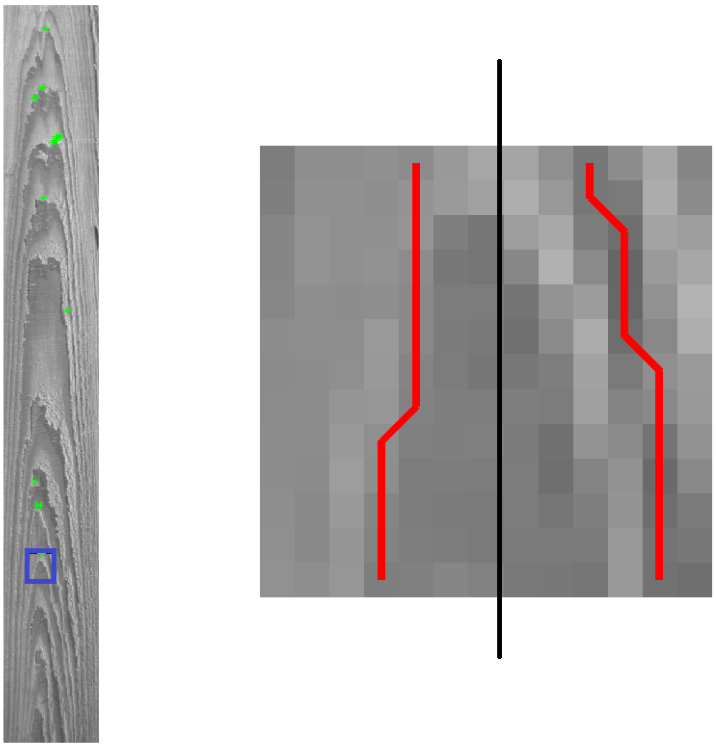


Figure 48 The left image is processed by dynamic programming (step 4). The image is going through a scanning and the green dots are the points where an upward wave direction is found. The blue window is the area which is cropped, resized to 25% and then split in the middle before being processed by Dynamic programming. The process of DP can be seen in the image to the right. The borders are finally traced by DP (red lines).

5.3.1.2 Results & discussion

The outcome of the method can be compared in Figure 50, when same image of the annual growth rings is turned upside down. The detections are less in the downward directed image because the blue window (in Figure 48) is placed directly under the point which is detected by integral image (step 3). It misses the wave which is downward directed. This can easily be corrected by placing the blue window over the point instead or one can do step 4 twice. First under the point and then over the point, see Figure 49.

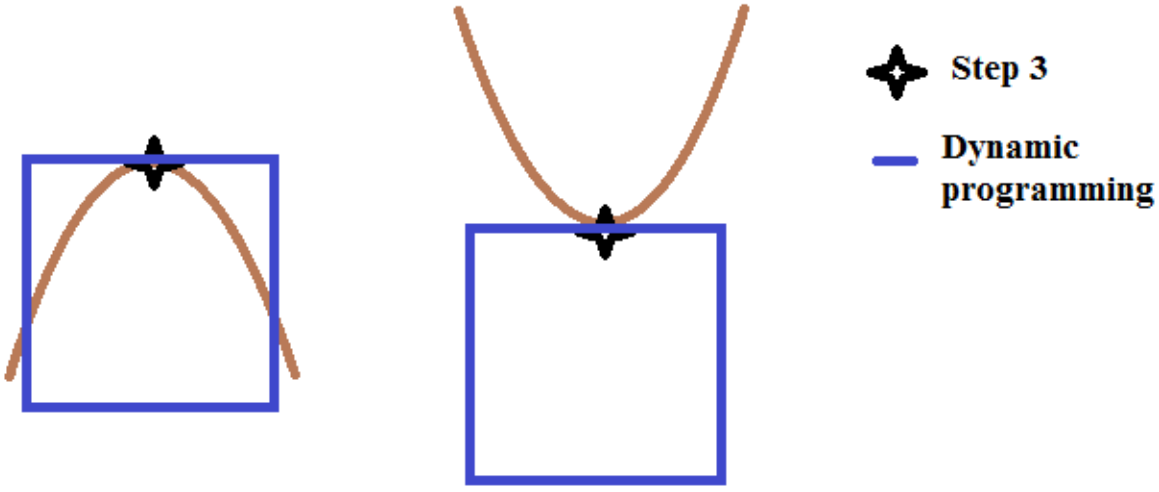


Figure 49 The window in dynamic programming is missing the downward directed wave. One can do the detection twice, onetime over the point and onetime under the point from step 3, to correct this.

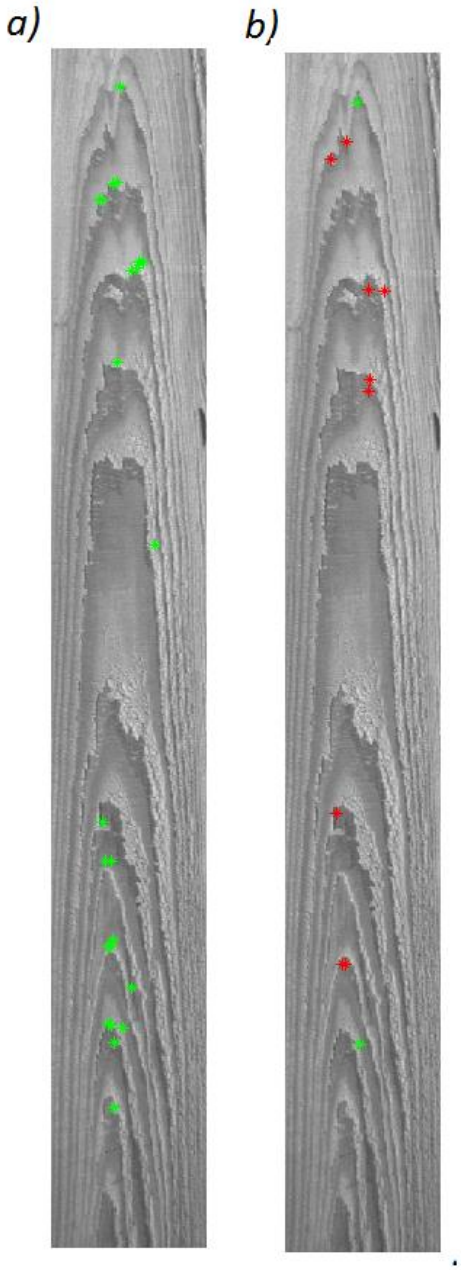


Figure 50 This is an example of the same image as in Figure 48 *but it's turned upside down (right) and scanned in the same process by step 3 and step 4. The green dots are the waves detected with an upward direction and the red dots are the waves detected with a downward direction. The image to the right is then turned back 180 degrees so that it can be compared with the left image. There were 23 detections made in the left image and 11 detections in the right image.*

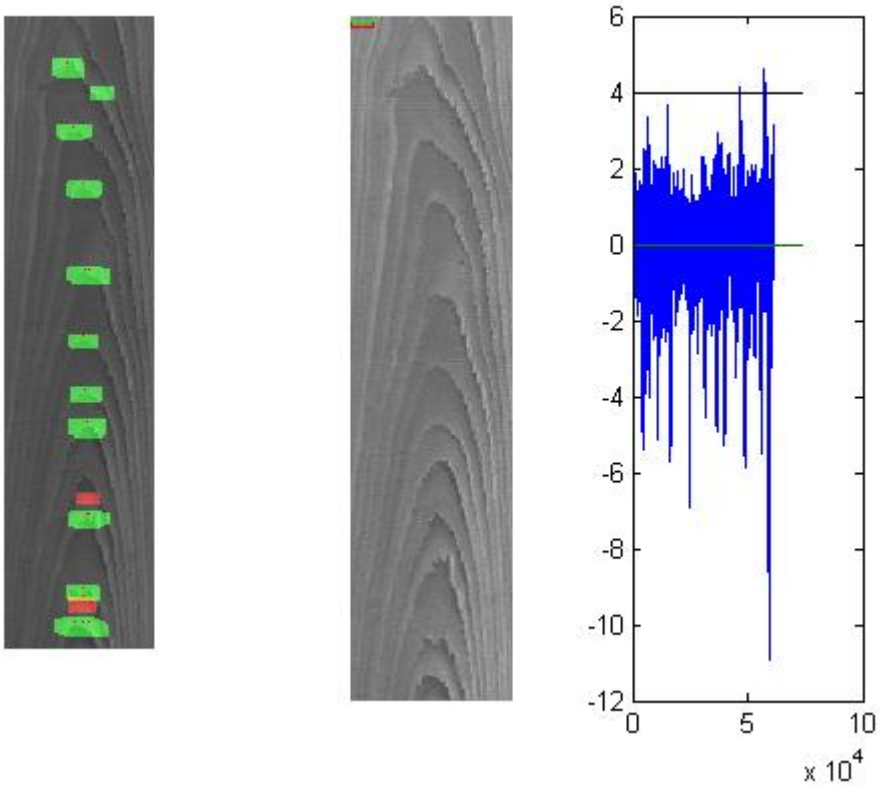


Figure 51 one more example of annual growth rings in step 3.

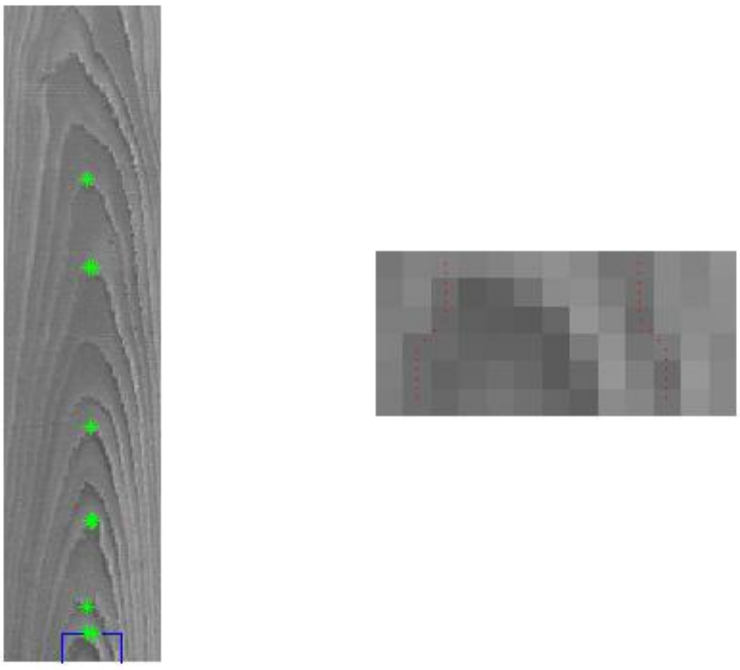


Figure 52: The outcome from step 4.

This feature which is extracted tells us if there is an upward, green dot, or down ward, red dot, pointing object. If there are many green dots placed almost along a straight path in a following order. Then it is with a high probability the pattern of annual growth rings. And by using this pattern one can prevent the classifier from misclassifying a healthy board. The performance results can be seen in *Table* 10.

5.3.2 Water stain detection

Water stains are easily confused with wood decay but one visual appearance that reveals a water stain is a completely straight border on the wood texture. This is the line which was created from a stick during a drying treatment. This border can be detected with the same method used in step 3 and then with a step 4 to extract the feature.

The method used in step 4 is a well known line detector used in computer vision, which is called Hough transform. One can reduce the computational time by using the same outcome from step 3.

However, sometimes the brightness difference between the two areas that contains healthy and water stained texture can be very small. This is because water stains can have a faint appearance. One can use a different size on the Integral image patch (step 3) and use a higher sensitivity to improve the detection of this border.

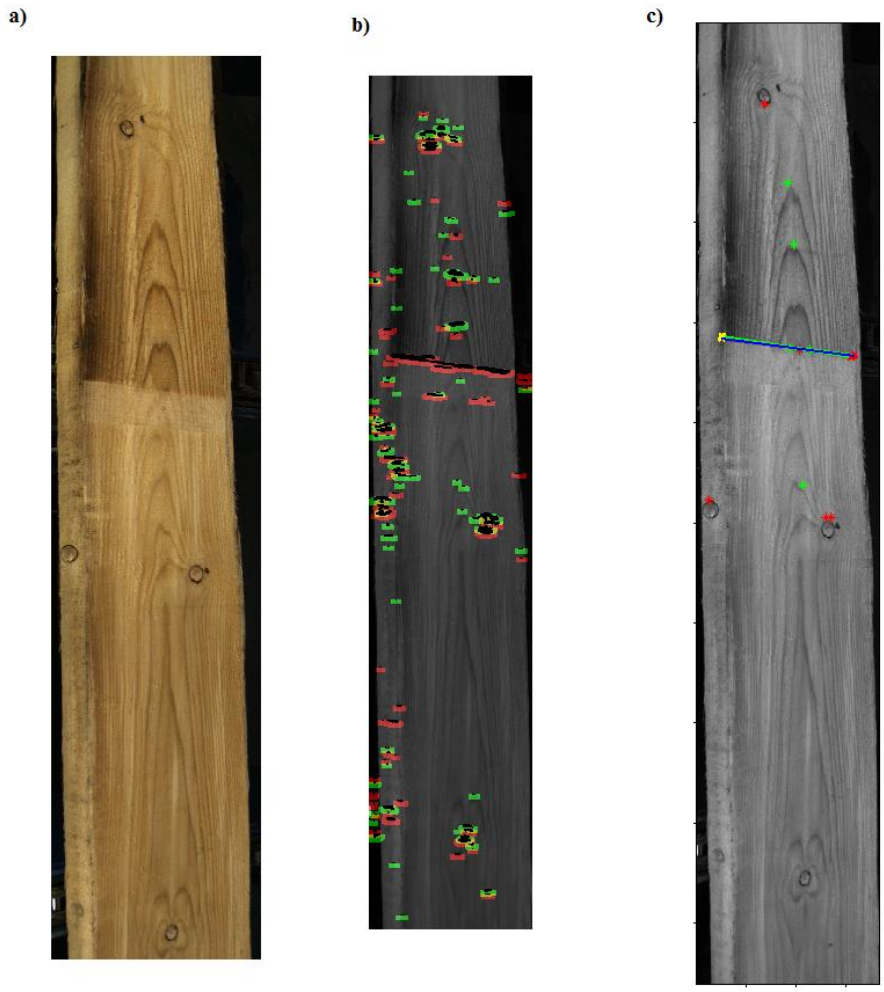


Figure 53: The original image, a), of one water stained board goes through each step. The outcome from step 3 can be seen in, b). The patterns are detected with Hough line detection, horizontal line, and dynamic programming, dots, in c).

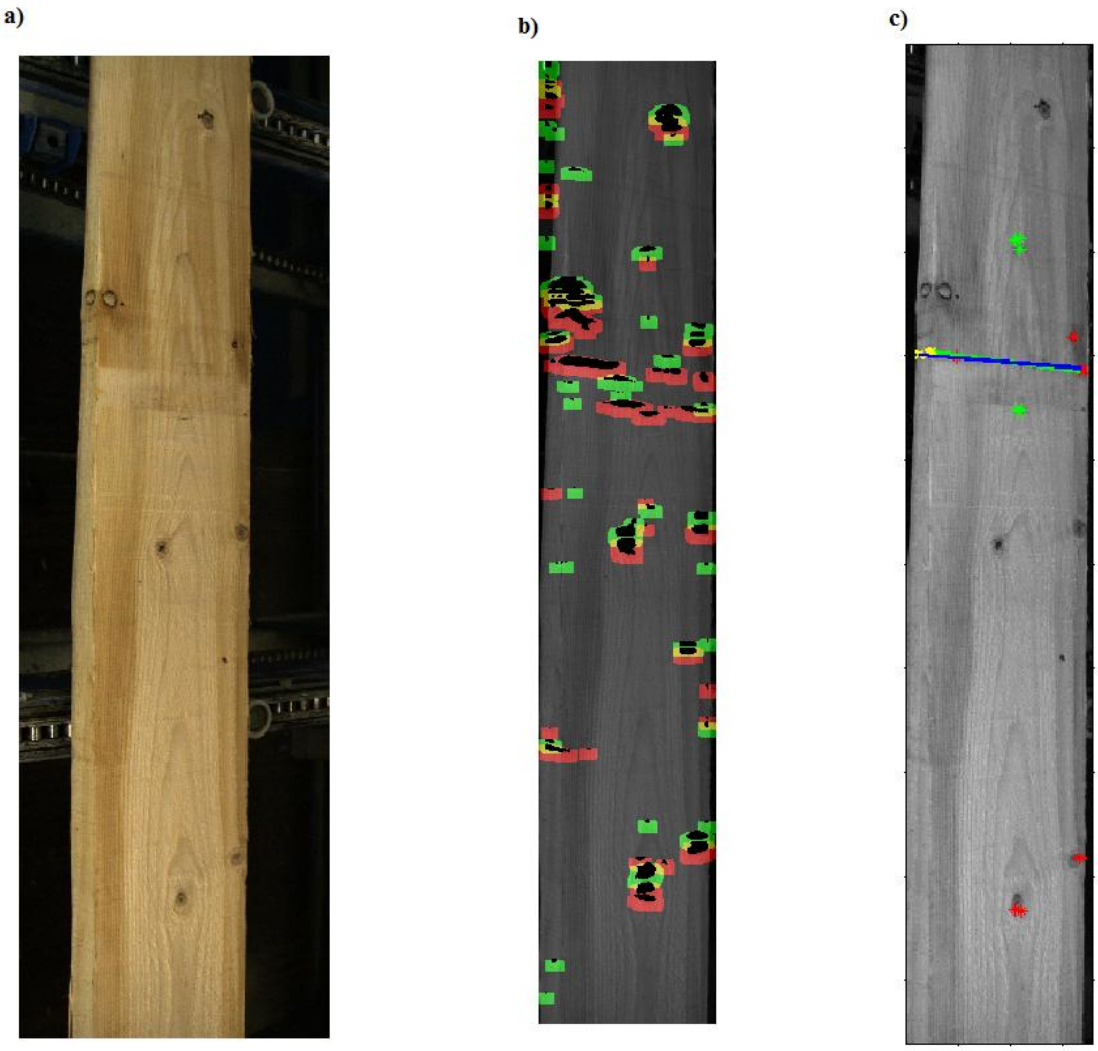


Figure 54 one more example of a water stained board.

For more test results from the program see Appendix.

5.4 Shakes (cracks)

5.4.1 Experimental setup

The methods used in detecting shakes are different from the ones used in rot and blue stains. The first method used is SUSAN, which can find edges on a gray level image. The sensitivity of SUSAN is controlled by two parameters. t in eq. (21), is the threshold for brightness difference between any pixel from the mask and the mask nucleus.

T is the threshold for the number of detections n within the mask. In this experiment a mask with the size 3x3 pixels is used.

The threshold values t=10 and T=5 works quite well for the board images I use.

5.4.2 Results & discussion

When the edges are found the pattern can be extracted with Hough line detector [Duda and Hart 1972]. Some examples can be seen in figures below.

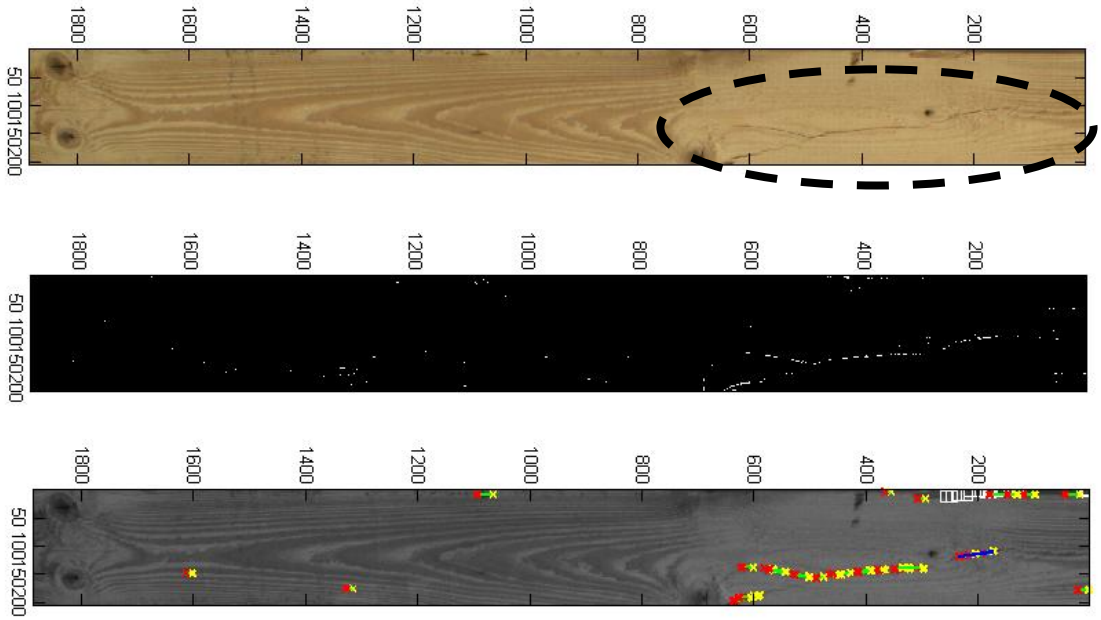


Figure 55 Top image shows a board that has shake defect (dashed). The middle image shows the outcome by using SUSAN. The bottom image shows the results of using Hough transform to detect lines so that the cracks can be traced.

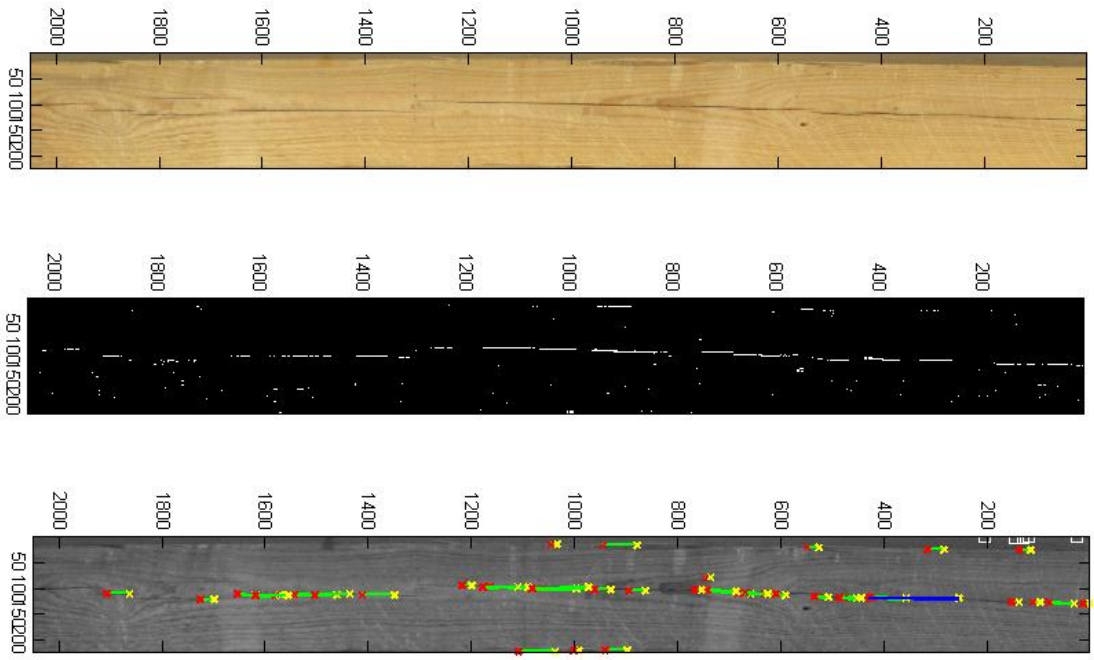


Figure 56

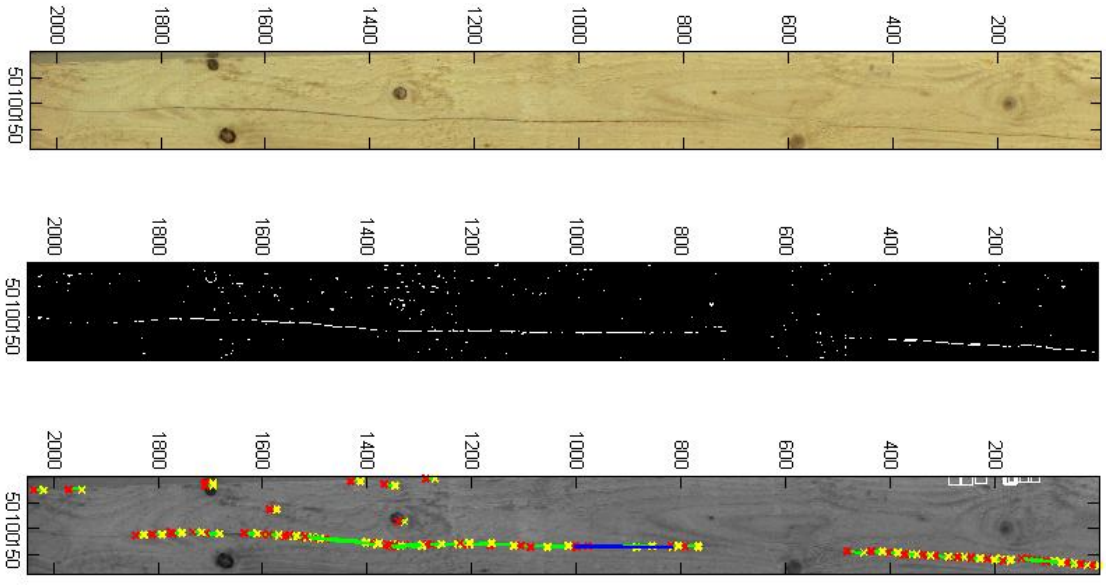


Figure 57

The detection of the shake pattern works with good accuracy but the rate of false detections is high. Some further feature extraction methods are required to exclude these false objects. The pith and the border edges, from the annual growth rings, are the common false objects. A typical example of false detection caused by the pattern of annual growth rings can be seen in figures below.

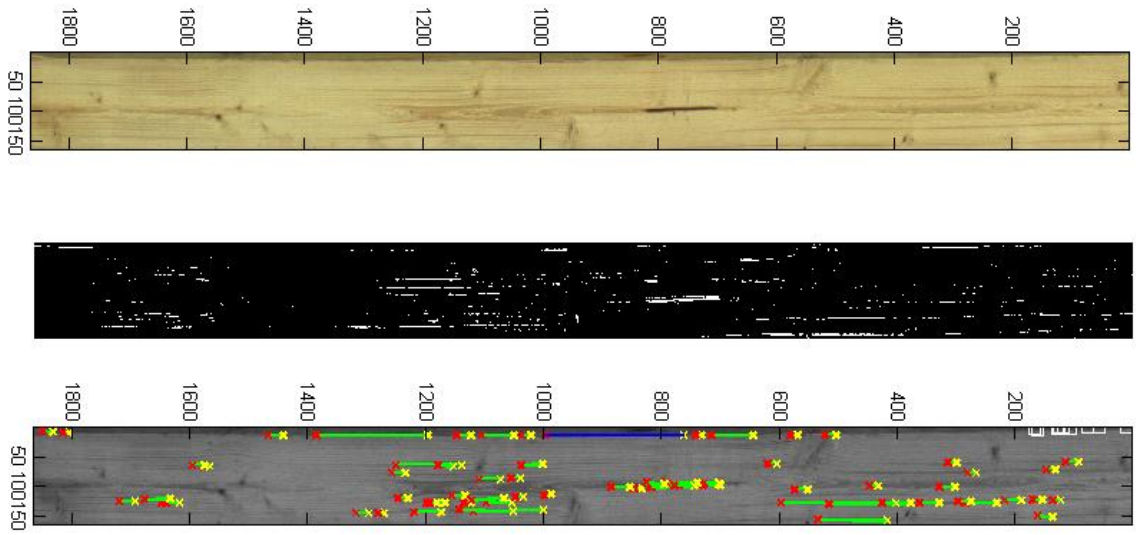


Figure 58: The lines from annual growth rings can be detected as false objects.

One can use a threshold on the edge pixels found by SUSAN to avoid the detection of annual growth rings. The annual growth ring is bright compared to a crack so by using a threshold one can exclude the edges that belong to bright pixels. The simplest way of avoiding false detection caused by the edge of the board is to actually exclude it from the searching area.

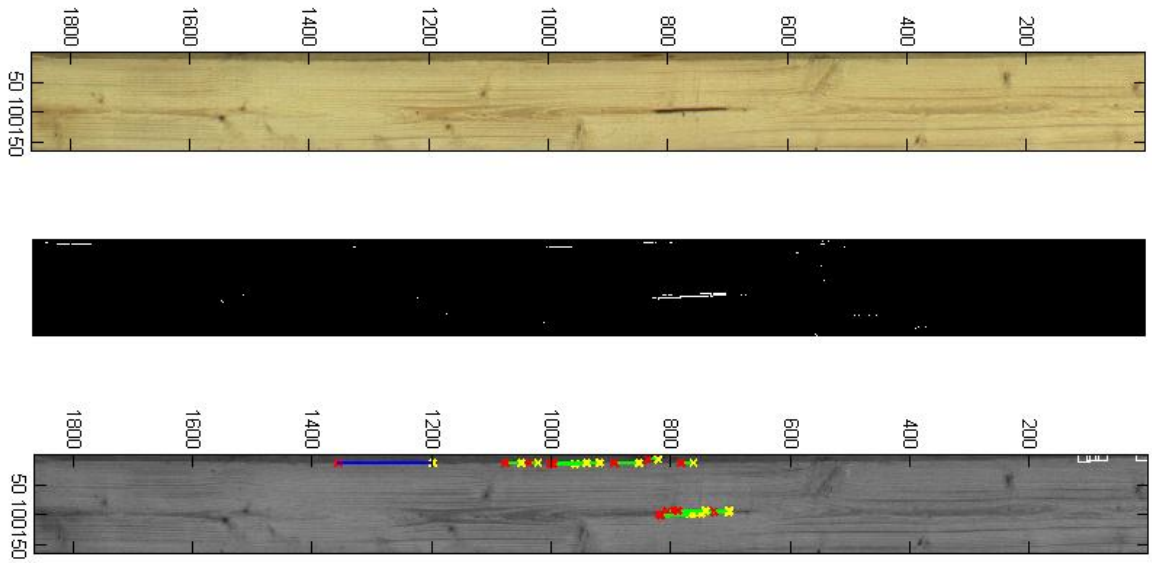


Figure 59 This is the same image used in Figure 58. *One can use a low threshold value to reduce many of the false lines detected in* Figure 58. *The threshold used only demonstrates how to avoid the lines from annual growth rings.*

The shakes in Figure 55, Figure 56 and Figure 57 are still detected with the same accuracy when using the threshold. If the threshold is to low then even the cracks will be neglected.

Now that the annual growth rings are avoided one type of false object still remains detected. The pith is to dark to exclude with a pixel intensity threshold.

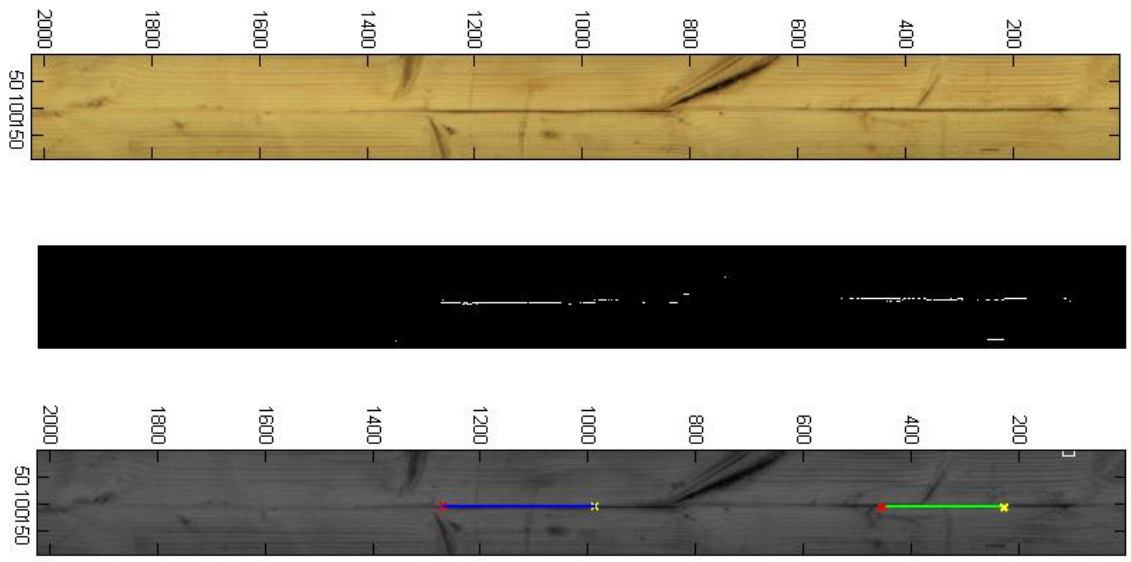


Figure 60 A pith can be seen on this board which is also detected.

The main property of the pith that one can use is its brown color and its straightness along the board surface. The pith line is not always continuous on the board.

One can exclude a pith line by further analyzing the detected lines. If the lines that are detected on a 10 m long board are aligned then it is most likely to be pith line.

I'm no expert in timber but in my opinion, the probability that a shake is straight throughout the board is very low. The brown color feature is a strong hint if the alignment argument is not strong enough. It is easier to see the brown color in reality and compare it to a dark shake. A narrow crack is very difficult to see with a camera if the pixel resolution is not sufficient and also if the light condition is bad. The camera captures the reflected light from the shake and stores it as a pixel value. But with a low resolution the dark crack will be smeared with the surrounding board colors, which will result in a shake with a color that is similar to the pith color. The performance results can be seen in *Table* 11.

5.5 Time budget

Time is an important factor in industry so the tools suggested in this thesis are supposed to be very time efficient. With one CPU core at 2,4 GHz it takes only 200 ms in Matlab to classify all pixels in a RGB image of 3,5 megapixels with one classifier table. If the same methods are programmed in C# then it is estimated that they will operate 10 times faster than in Matlab.

The integral image is created with a nested for-loop in Matlab which is considered to be very slow but the time it takes to complete one integral image is 50 ms. The dynamic programming requires heavy computation when the input image is large. In the test results the image size of 156 pixels, blue window (see Figure 48), is used to minimize the computation time. The time to complete one blue window in DP is 2 ms.

The total time in scanning an image with DP is estimated to be 2 ms multiplied by the number of times DP is used. To avoid scanning the same area multiple times with DP, it is highly recommended to use a function that locates a central point in a cluster. The points detected with integral image (step 3) can be very redundant.

The Hough transform computation time depends entirely on the step resolution. The Hough line detection is a method that detects if pixels are aligned by letting it draw lines at different angles and locations on an edge image. For each line it draws it counts all pixels that are found on that line.

This is a very robust computation that tries all possible lines and the time depends most on how many lines that are tried out. There is a tradeoff between the computation time and precision. A finer Hough transform step resolution will result in a high precision and a long computation time.

The SUSAN edge detector is used with a mask size of 3x3 and an image of 1,2 megapixels. The image is scanned with SUSAN at 2,4 s. The long computation time is directly related to the slow Matlab for-loop.

If the same methods are programmed in a fast computer language let's say C and if the code is programmed more cost-effective, then it would take only a fraction of a second to complete the image with the same CPU speed.

Table 9 The processing time in Matlab with one CPU core at 2,4 GHz

Program	Estimated time in Matlab (s)
To classify all pixels in an RGB image of 3,5 mega pixels with a classifier table	0,2
Integral image on 3,5 mega pixels	0,05
Dynamic programming on an image with 156 pixels	0,002
Hough transform- The time it takes to complete one image of 3,5 mega pixels in my program.	(Depends on the resolution) 0,5
SUSAN on a GL image of 1,2 mega pixels	2,4

5.6 Performance evaluation

The test images used in the performance evaluation of shake and annual growth ring detection were handpicked by me. I am not so familiar with wood defects but I chose from what I recognized as a defect. For the evaluation of rot detection, I had 100 images that were handpicked by someone else who is more experienced with rot than me and can distinguish rot from water stain. 50 images contain rot and the other 50 images contain objects that can be confused with rot e.g. water stains. I also tested the performance of blue stain detection on 35 images that contain boards with blue stains.

5.6.1 Annual growth rings

Table 10 10 images where used, which contains approximately 77 waves from the annual growth rings.

Waves	**Detections**	**Detection rate**
77	55	71,4%

5.6.2 Shakes (cracks)

Table 11 Crack detection from 10 images containing 106 cracks, the edges of the board was not included in the search. The pith was also excluded from the test.

Shakes(cracks)	Detections	False detections	Detection rate	False detection rate
106	86	9	81,1%	8,5%

5.6.3 Rot

When searching for rot with gray and brown color.

Table 12 50 board images containing rot samples where used.

Rot samples	Detections containing gray	Detection rate of gray color when brown is detected
54	36	66,7%

Table 13 50 board images where used containing samples of objects that can be confused with rot and which causes false alarms. These objects are usually annual growth rings and water stains.

False object samples	Detections containing gray	Detection rate of gray color when brown is detected
59	37	62,7%

Table 14 Data statistics: proportion of gray color within brown, 36 samples of rot & 37 samples of false objects

Type	Mean	Std	Median
Rot	0.0944	0.0892	0.0483
False objects	0.072	0.101	0.033

5.6.4 Blue stain

Table 15 35 images of boards that have blue stains where used.

Blue stains	**Detections**	**False detections**	**Detection rate**	**False detection rate**
54	42	4	77,8%	7,4%

6 Future Work

With the methods used and proposed in this work, it is possible to detect the defects and also in addition to detect some of the false objects. By detecting the false objects one can increases the accuracy of the classifier.

The test results show that the detection of "annual growth rings" and "water stains" works as long as the repeating waves and the stick mark are visible on the board. To further improve the detection accuracy of wood decay, it is necessary to be able to detect some more types of false objects or, put another way, extract the features which can be associated with the false objects.

There are some features that can be further analyzed and exploited with image analysis methods. E.g. the water stain problem is not always visible with the straight border so some other features are required to recognize water stains.

It might be possible to use multiple classifiers where each classifier is trained to identify a different shade of brown color in water stains as in *Figure 61*. The difficult part in doing that is to train the classifier to be sensitive enough. As the samples of the different brown colors are too close to each other the training with Adaboost will result in a classifier which is too sensitive to small color changes. The classifier will misclassify pixels if small changes occur in the ambient light because of the narrow color range.

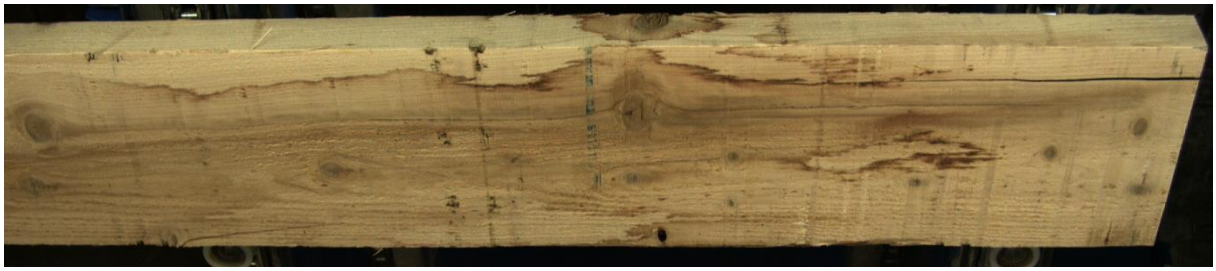


Figure 61 This board has water stains on it. One feature that is clearly visible is the darker brown color that is found at the borders.

With dynamic programming it was possible to trace the wave pattern from the annual growth rings. There are some other visible patterns that can be extracted and exploited by using image analysis methods. One of them is the pattern of the slim uneven lines that appear often, see Figure 62. The color of the lines will sometimes be detected as rot so to avoid detection. The search program should extract the pattern so that it can be analyzed and identified as a false object. The lines can be detected with the line detector (Hough transform). The test results from shake detection proved this because the straight lines of the annual growth rings where interfering the search for shakes. They were avoided with a threshold.

Figure 62 The growth rings appear almost as straight lines (black window). Hough transform can extract these lines. And if a pattern is recognized it can be used to increase the accuracy of a classifier.

.

7 Conclusion

In this thesis several methods are studied, combined and tested to see if they can be used to develop a program that can automatically detect defects on timber and meet up to the time efficiency requirements in sawmills.

The test results show that there are potentials in using machine vision to aid sawmills. The defects that cover large areas are detected first by their colors with a supervised classifier. The classifier is created with the boosting algorithm Adaboost.

It is then suggested to use some feature extraction methods to rule out all false objects which can have the same color as the defect.

The detection of cracks uses a different kind of image analysis approach because of their shape which is visible with few pixels. The detection of shakes is done first by using an edge detection method, SUSAN, and then a line detection method, Hough transform. The challenge does not lie in detecting the lines. It is to distinguish the cracks from the false objects that are also detected with the same methods.

The detection of cracks is more affected by noise than the detection of rot and blue stain defects. The SUSAN edge detector is quite resistant toward noise. One can detect cracks that are very faint by increasing the sensitivity of SUSAN. But as the edge detector's sensitivity increases the noise will also increase.

The classifiers that are trained to identify a defect's colors are sensitive to changes in the ambient light. These color detecting classifiers are used in combination with integral image to focus on regions of interest in a fast way. These regions are further analyzed with suitable methods so that features can be exploited to indicate if the object is a defect or a false alarm.

- Blue stain
    - Is detected by using Adaboost and a weaklearner to identify the unique colors of blue stain. The detection works accurately as long as the search area on the board is faced towards the camera.
- Wood decay(Rot)
    - Is found by its color with the same method used in detecting Blue stain but the color is not unique so the false objects which is detected have to be separated from the true rot. This problem is very complex to solve with only information which is provided from images taken in the visible spectrum. The rot does not have any form or shape that can be predicted. I suggested methods to work around this problem by searching for rot indirectly. The strategy is to use methods that can extract patterns from false objects, objects that can be recognized from their properties. The false alarms can be reduced by actually identifying the two most likely false objects.

- Shake (crack)
    - Is detected by using an edge detector like SUSAN and then use Hough transform, which is a line detection algorithm. Many false objects can be excluded by applying an intensity threshold so that bright pixels will be neglected. One common false object, the pith, will remain detected even if a threshold is used. A method is proposed on how to recognize this type of false object, which actually is a defect.

8 References

Crow, F.C. (1984): Summed-area tables for texture mapping, SIGGRAPH Computer Graphics, Vol. 18, No.3, {July} 1984, pp. 207-212

Duda, R.O., Hart, P.E. (1972): Use of the Hough Transformation To Detect Lines and Curves in Pictures, *Communications of the ACM,* Vol. 15, No. 1, {January} 1972, pp 11-15

Forest Products Laboratory (FPL) (1999), *Wood handbook- wood as an engineering material*. Gen. Tech. Rep. FPL-GTR-113. Madison, WI: U.S. {1999}. 463 pp.

Freund, Y., Schapire, R.E. (1997): A decision-theoretic generalization of on-line learning and an application to boosting, *Journal of Computer and System Sciences*, Vol. 55, {August} 1997, pp. 119–139

Föreningen svenska sågverksmän (FFS) (1999): *Nordisk trä* {Nordic tree}, FFS, FS and TTF, {2}, {sweden}, {1999}, 64 pp.

Kandel, ER., Schwartz, JH., Jessell, TM. (2000): *Principles of Neural Science (4th ed.)*, McGraw-Hill, New York, U.S. 2000 pp. 182-185

Knaebe, M. (2002): Blue Stain. *Techline,* {May} 2002, pp. II-2 - 2

Kuncheva, L.I. (2004): *Combining Pattern Classifiers, Methods and Algorithm,* Wiley-Interscience, {July} 2004, 376 pp.

Mallick, D., Lee, V.C., Yew Soon Ong (2008): An emperical study of Genetic Programming generated trading rules in computerized stock trading service system. *Service Systems and Service Management, 2008 International Conference on*, {August} 2008, pp. 1-6

McCarthy, J. (2007): What is Artificial Intelligence? *Stanford University* , {November} 2007

Meir, R., Gunnar, R. (2003): An introduction to boosting and leveraging, *Advanced Lectures on Machine Learning, LNCS, Conference*, pp.119-184

Papageorgiou, C.P., Oren, M., Poggio, T. (1998): A general framework for object detection, *ICCV '98 Proceedings of the Sixth International Conference on Computer Vision,* pp. 555

Schapire, R.E. (2001): The Boosting Approach to Machine Learning: An Overview, *MSRI Workshop on Nonlinear Estimation and Classification*, Berkeley, CA {March 2001}

Schapire, R.E, and Singer, Y. (1999): Improved boosting algorithms using confidence-rated predictions, *Machine Learning*, Vol. 37, No. 3, {December} 1999, pp. 297–336

Smith, A.R. (1978): Color Gamut Transform Pairs, *SIGGRAPH 78 Conference proceedings*, {August} 1978, pp. 12-19

Smith, S.M., Brady, J.M. (1997): SUSAN- A new approach to low level image processing, *International Journal of Computer Vision,* Vol. 23, No. 1, {May} 1997, pp. 45-78

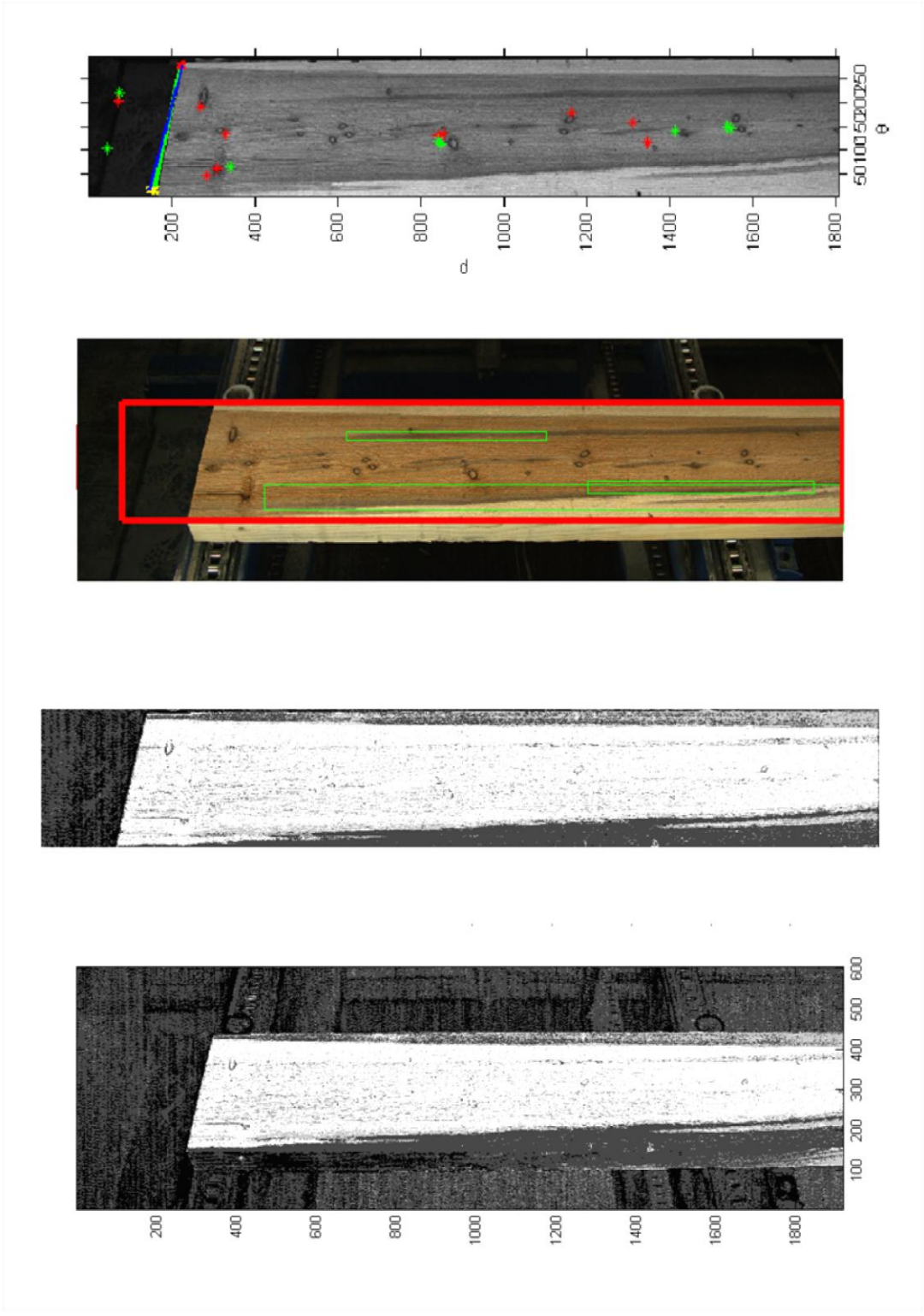
Sonka, M., Hlavac, V., Boyle, R. (2007): *Image Processing, Analsysis and Machine Vision,* CL-Engineering, third edition, {March} 2007, 800 pp.

Viola, P., Jones, M. (2001): Rapid object detection using a boosted cascade of simple features, *Accepted Conference on Computer Vision and Pattern Recognition,* 2001

Appendix A

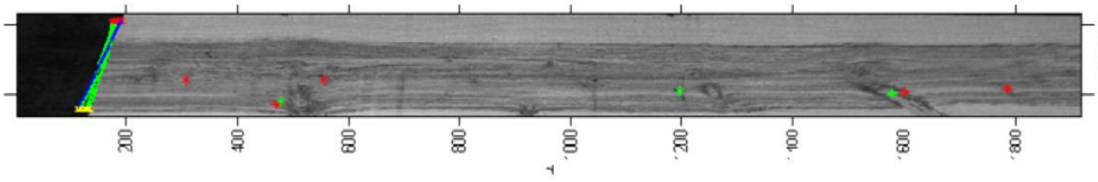
Some extra test results from the written program.

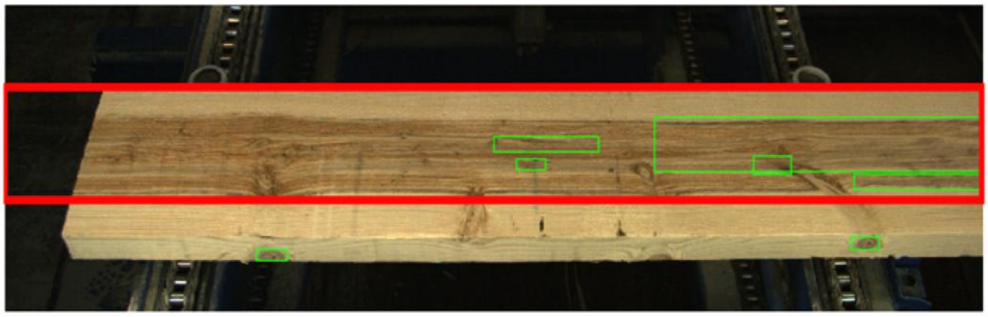
Wood Decay

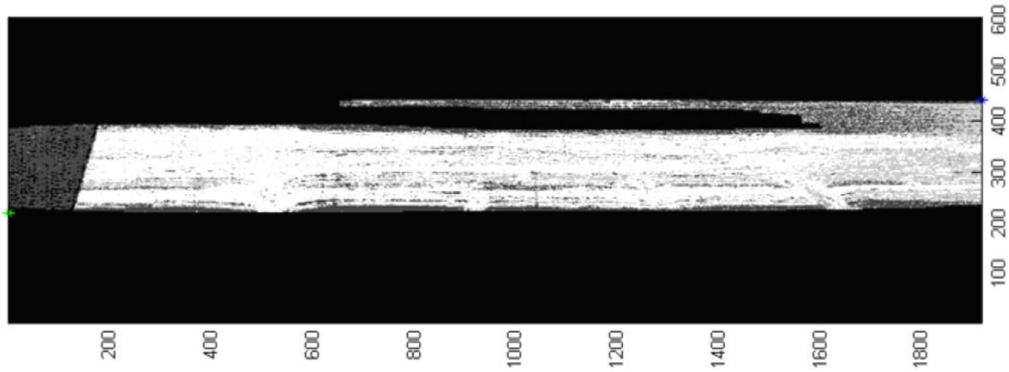


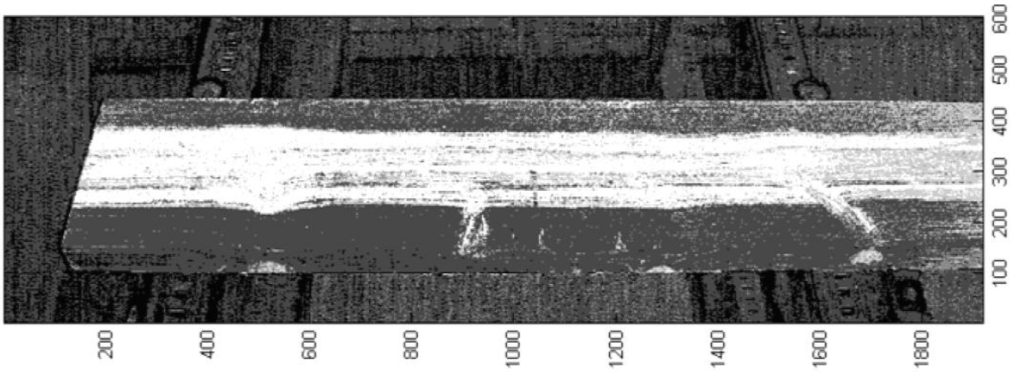
200
400
600
800
1000
1200
1400
1600
1800
ρ
θ
100
200
300
400
500
600

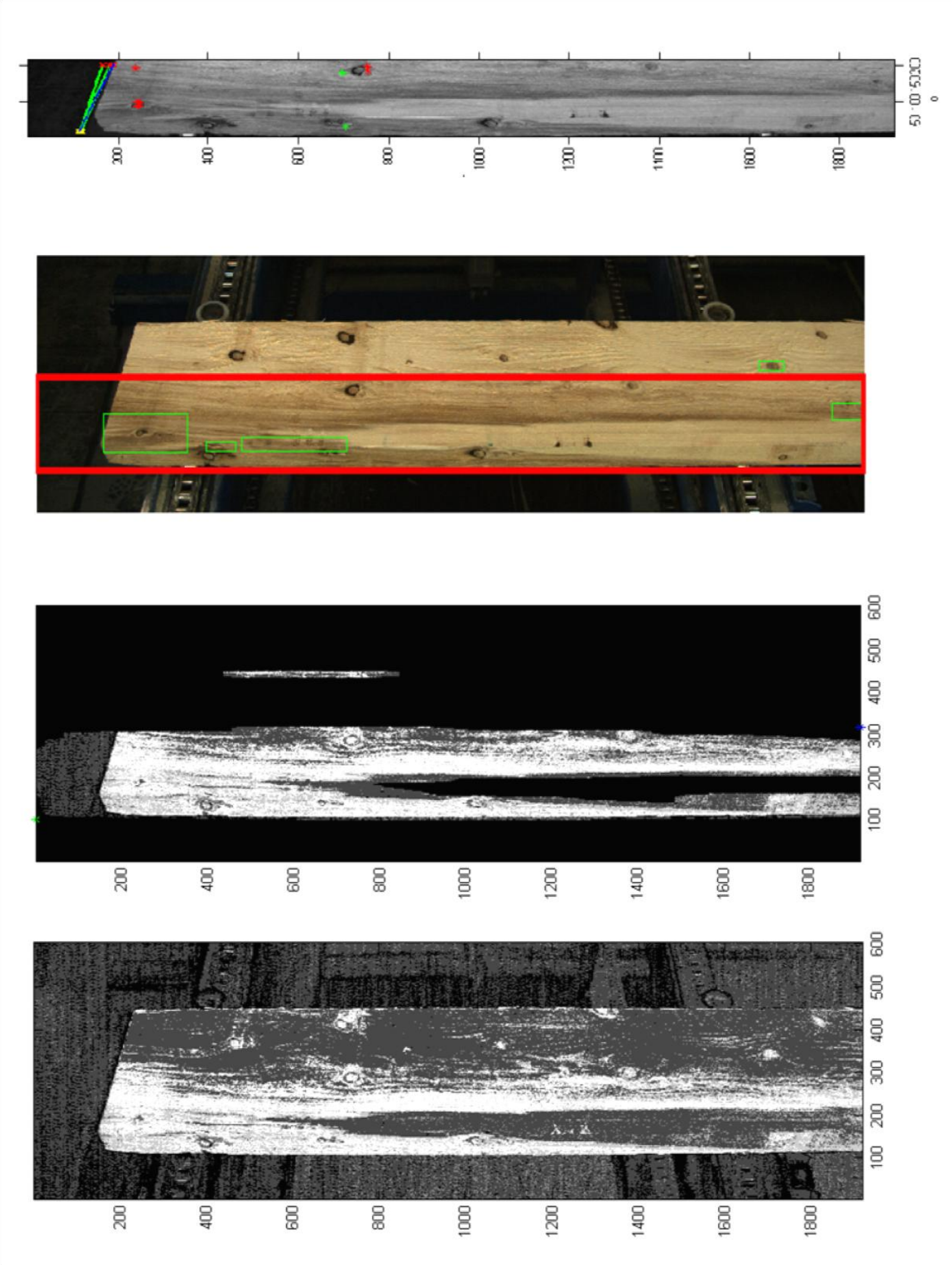
200
400
600
800
1000
1200
1400
1600
1800
100
200
300
400
500
600










200
400
600
800
1000
1200
1400
1600
1800
100
200
300
400
500
600

200
400
600
800
1000
1200
1400
1600
1800
p
100
200
300
400
500
600
e

200
400
600
800
1000
1200
1400
1600
1800
5010050
θ
100
200
300
400
500
600

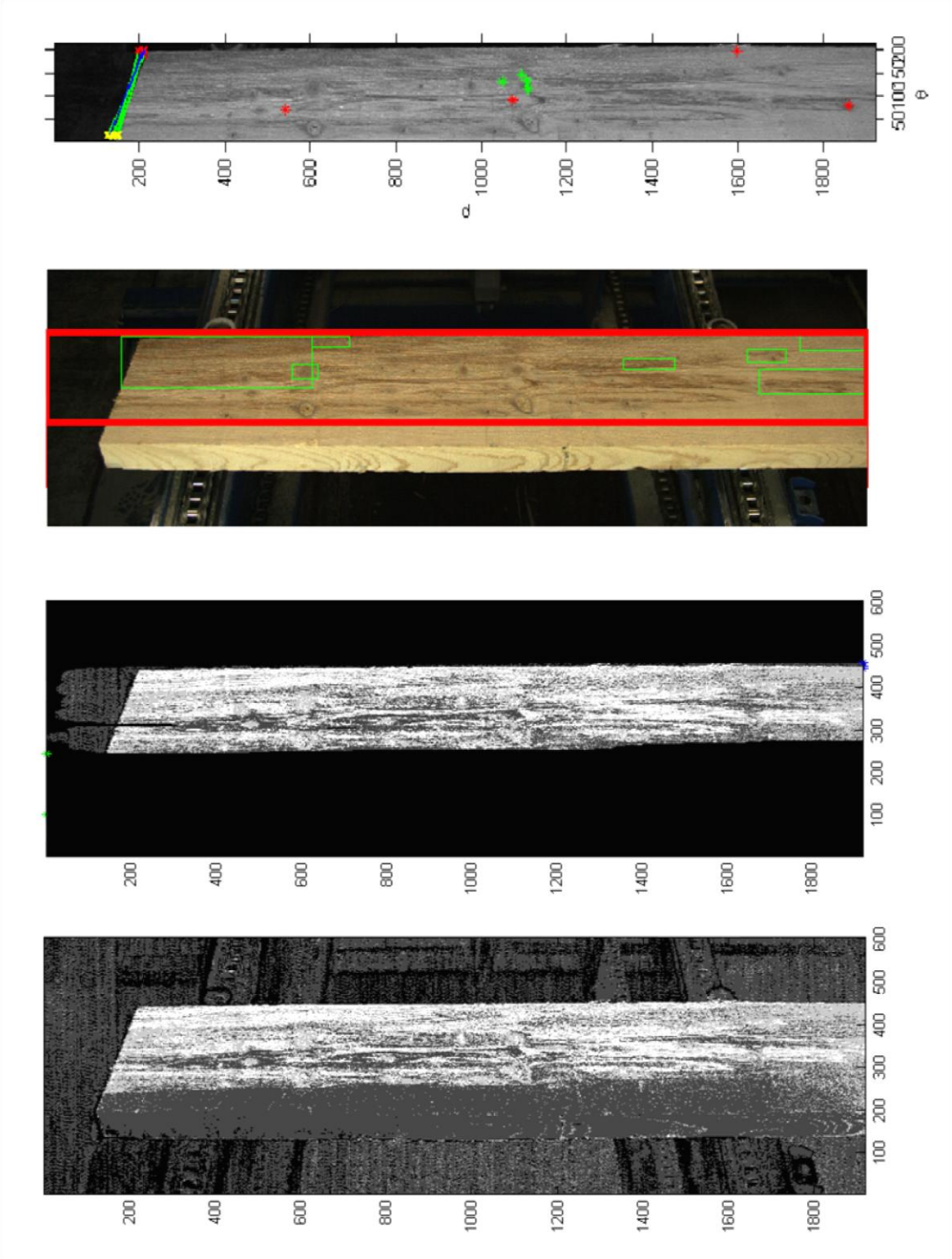
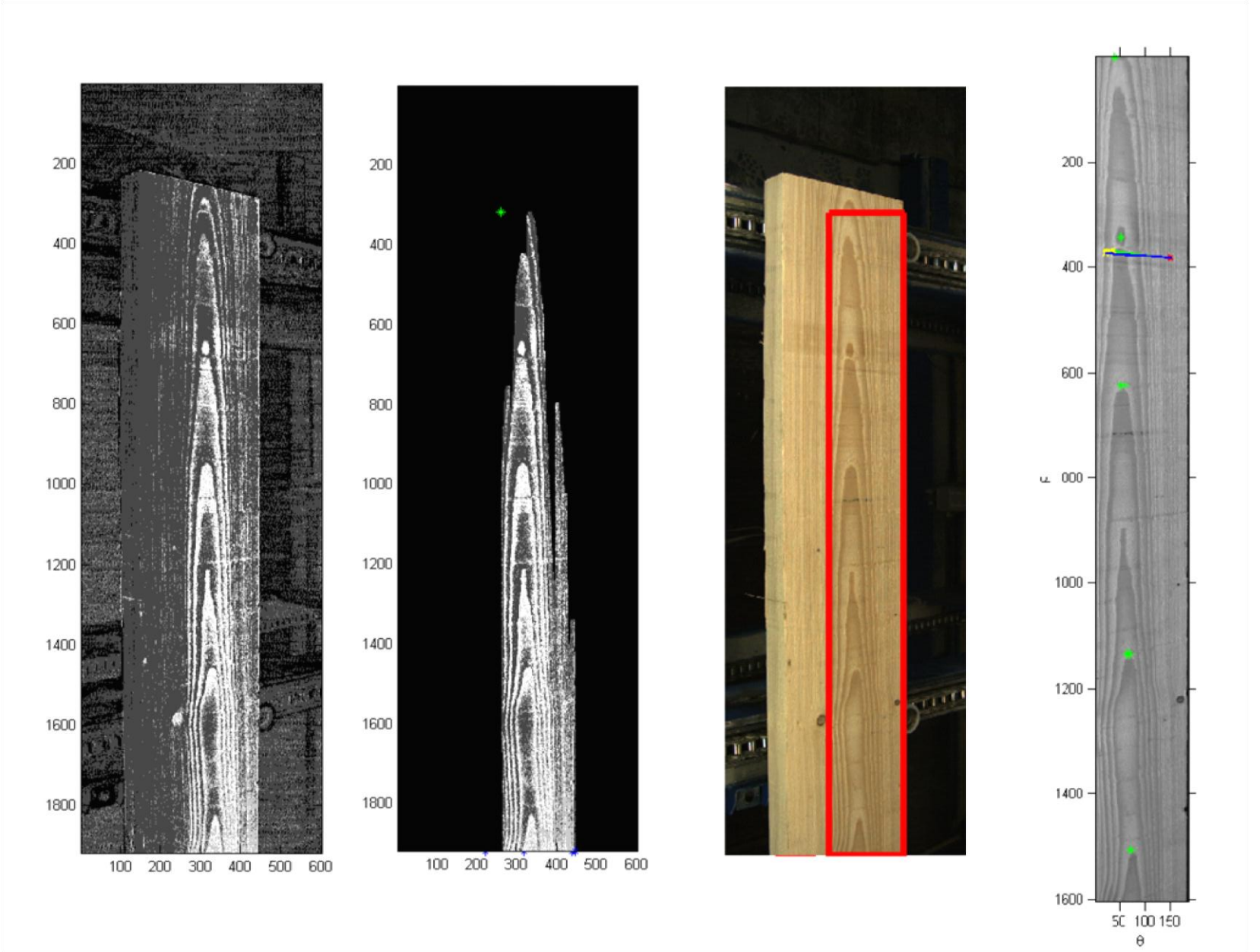
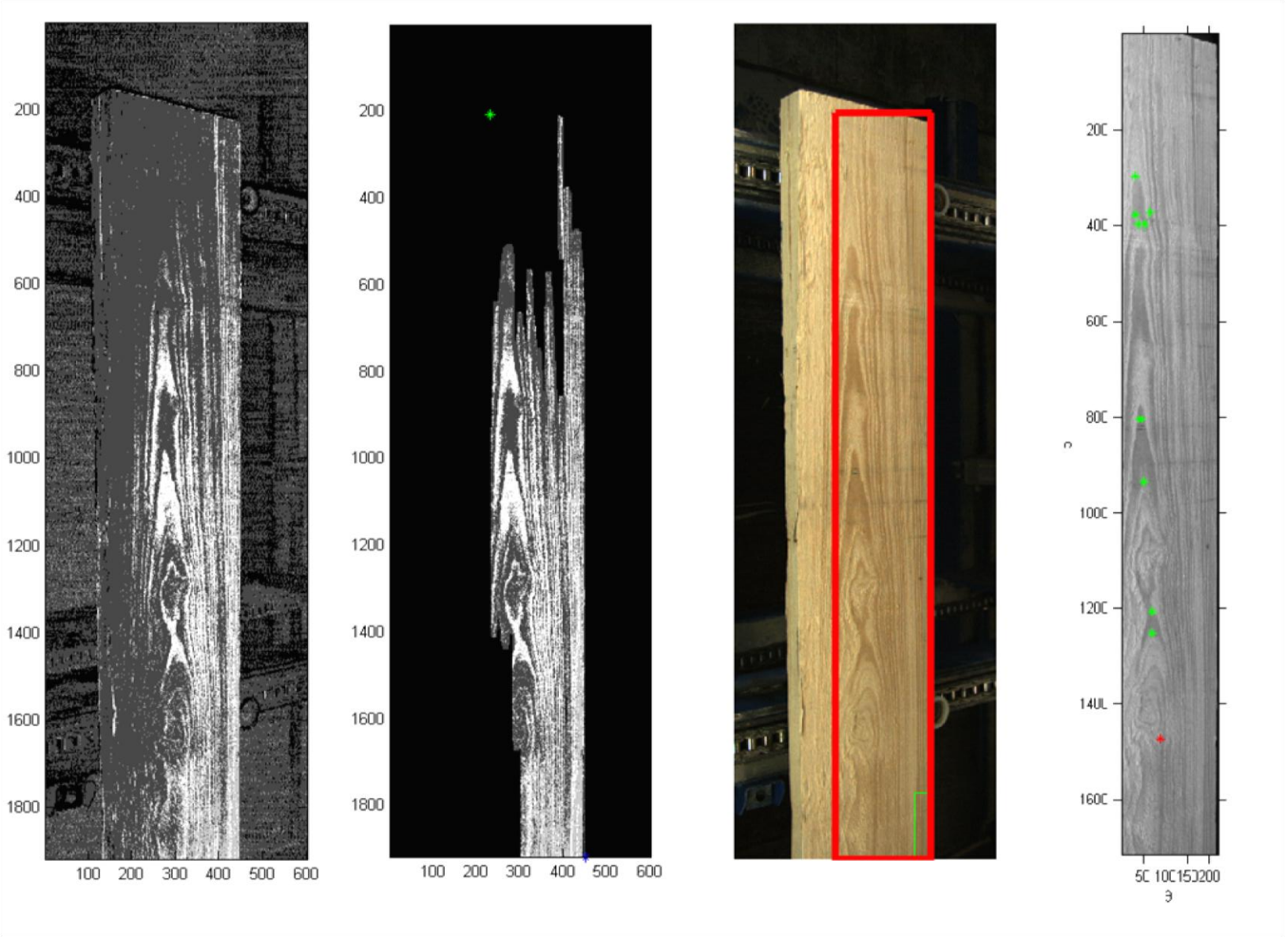


Figure 63 The three first images from bottom represents step 1 and step 2 while the image on the top is the result of step 3 and step 4. The red frame indicates the area which has the color of rot and the green frames indicates the area that contains the extra gray-like color. Don't mind the lines detected at the edge of the board. The green and red dots are scattered over the board surface and they do not appear as they are in order.

Water stains and false alarms


200
400
600
800
1000
1200
1400
1600
1800
100 200 300 400 500 600
200
400
600
800
1000
1200
1400
1600
1800
100 200 300 400 500 600
200
400
600
000
1000
1200
1400
1600
50 100 150
θ


200
400
600
800
1000
1200
1400
1600
1800
100 200 300 400 500 600
200
400
600
800
1000
1200
1400
1600
1800
100 200 300 400 500 600
200
400
600
800
1000
1200
1400
1600
5C 10C15J200

200
400
600
800
1000
1200
1400
1600
1800
0
50
100
150
200
250
θ
200
400
600
800
1000
1200
1400
1600
1800
100
200
300
400
500
600
200
400
600
800
1000
1200
1400
1600
1800
100
200
300
400
500
600

200
400
600
800
1000
1200
1400
1600
100
200
300
400
500
600
1800

200
400
600
800
1000
1200
1400
ρ
50
100
150
θ

100
200
300
400
500
600
200
400
600
800
1000
1200
1400
1600
1800

100
200
300
400
500
600
200
400
600
800
1000
1200
1400
1600
1800

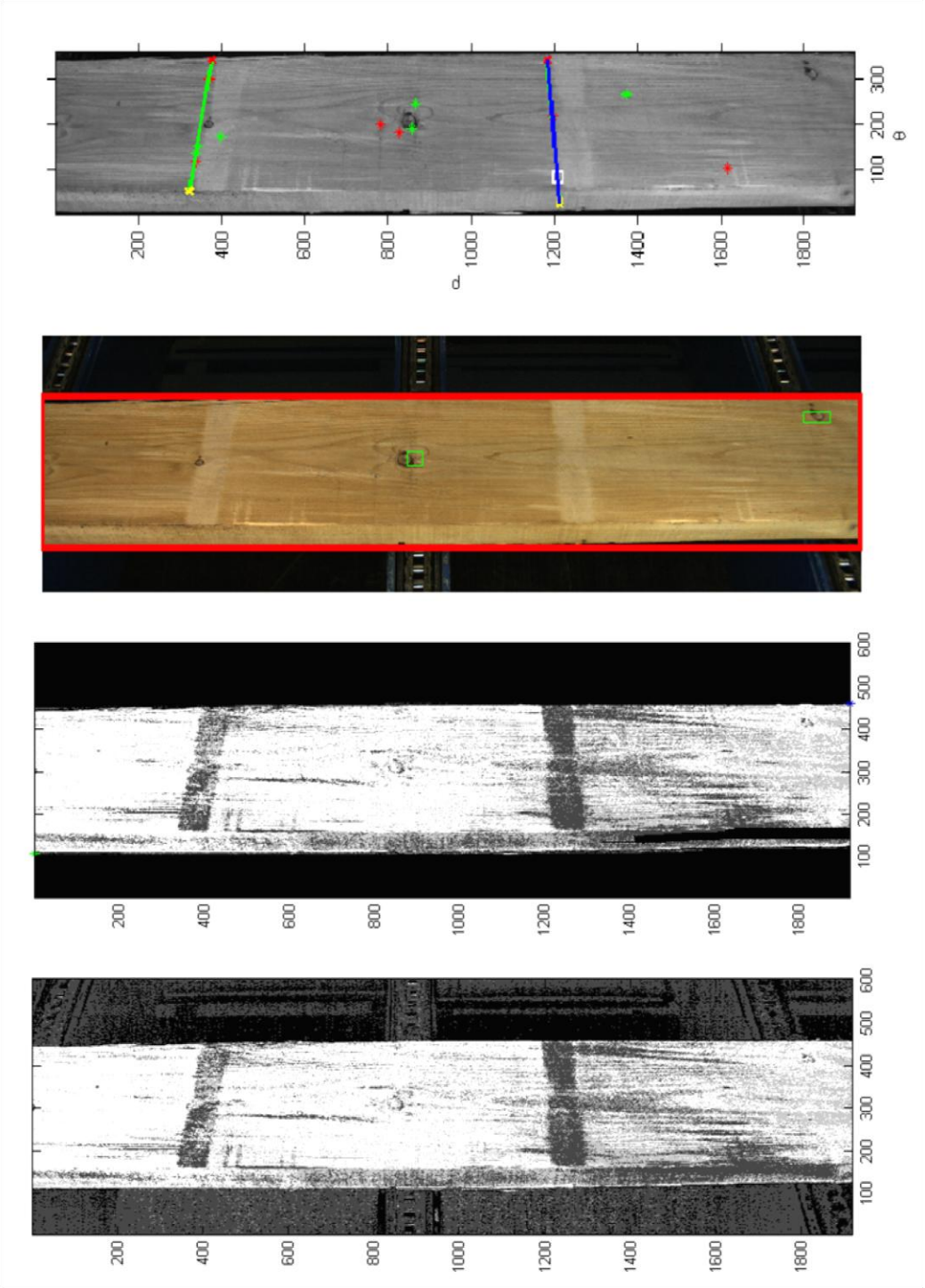


Figure 64 The green and red dots are more organized and they follow a pattern which is expected for annual growth rings. The lines detected across the board indicate that the detected area is actually water stain. These images are the false alarms when searching for rot but once the features and patterns are detected the false alarms can be neglect.